TPC Benchmark C®
Full Disclosure Report

Compaq AlphaServer ES40
Model 6/667
4 CPU
Client/Server System
Using Compaq Tru64 UNIX®
and
Compaq ProLiant 1600
Using
Microsoft NT V4.0 ®
and
Sybase Adaptive Server Enterprise® 11.9.3

Company Name	System Name	Database Software	Operating System Software
Compaq Computer Corporation	**Compaq AlphaServer ES40 Model 6/667 4 CPU**	**Sybase Adaptive Server Enterprise 11.9.3**	**Compaq Tru64 UNIX V4.0G**

Availability Date: March 17, 2000

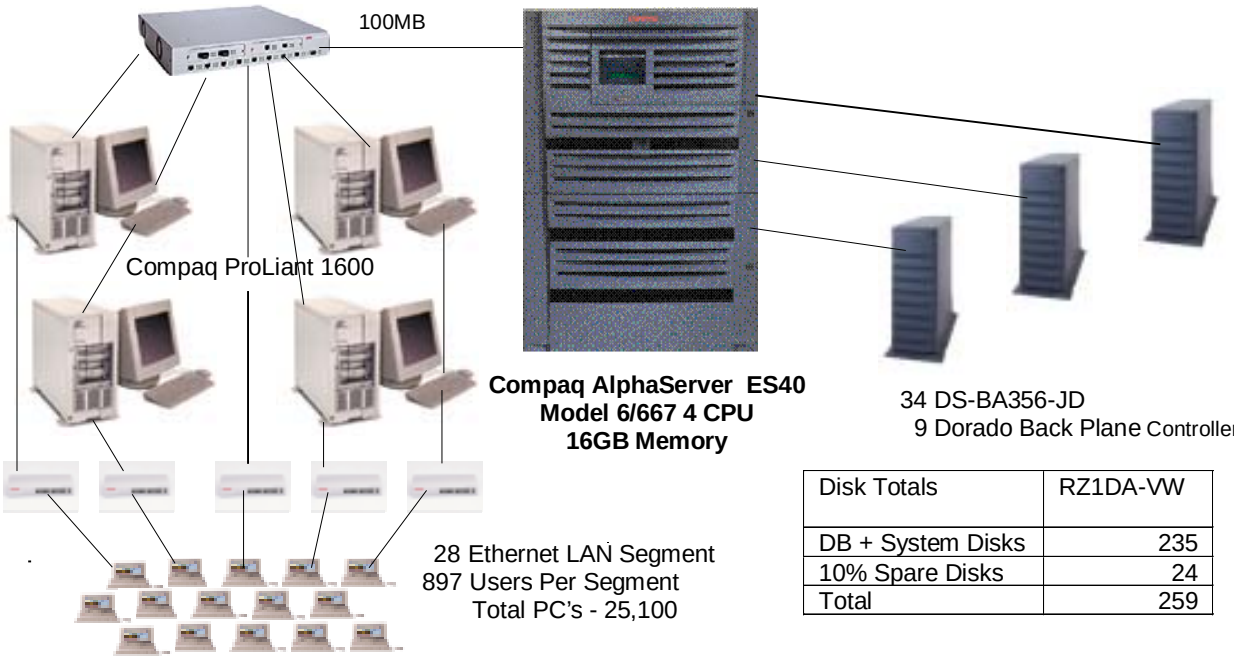
Total System Cost	TPC-C Throughput	Price Performance
- Hardware - Software - 5-Years Maintenance	Sustained maximum throughput of system running TPC Benchmark C expressed in transactions per minute	Total system cost/ TPC-C® throughput **$906,225/$29.48**
$906,225	**30,738**	**$29.48**

First Printing February, 2000

COMPAQ	Compaq AlphaServer ES40 Model 6/667 4 CPU Client Server System	TPC-C Rev. 3.4 Report Date: **February 9, 2000**

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$906,225	**30,738**	**$29.48**	**March 17, 2000**

Processors	Database	Operating System	Other Software	Number Users
4 667MHz Alphachip 21264	Sybase Adaptive Server Enterprise 11.9.3	Compaq Tru64 UNIX V4.0G	Application Optimizer WNT Server Edition	25,100



Disk Totals	RZ1DA-VW
DB + System Disks	235
10% Spare Disks	24
Total	259

System Components	Server Qty	Server Type	Client (each of 4) Qty	Client Type
Database Nodes	1	Compaq AlphaServer ES40	1	Compaq ProLiant 1600
Processors	4	667 MHz Alphachip 21264	1	600 MHz Intel
Cache Memory	8	MB		512KB
Memory	4	4 GB	1	128MB
Total Memory	16	GB	2	256MB
Disk Controllers	9	Backplane Raid Controller		
Disks	235	9.1GB Disks		
Total Disk Storage	2138.5	GB		

COMPAQ	AlphaServer ES40 Model 6/667 4 CPU C/S System	TPC-C REV 3.4 EXECUTIVE SUMMARY PAGE 2 OF 2 Report Date: February 2000

Description	Part Number	Third Party	Unit Price	Qty	Extended Price	5 yr. Maint. Price
Server Hardware		**Brand**	**Pricing**			
Compaq AlphaServer ES40 Model 6/667	DA-64AAA-EA	1	48,803	1	48,803	19,165
ES40 Tower Enclosure	BA61M-MT	1	350	1	350	0
ES40 SMP CPU; 667 MHz;UNIX	KN610-BB	1	8,750	3	26,250	0
ES40 2GB Memory Option	MS610-EA	1	19,354	7	135,47	37,884
ES40 Power Supply	H7906-A9	1	875	2	1,750	0
3 Chan. LVD 64Bit PCI Raid Controller 16MB Mem	DS-KZPCC-CC	1	2,660	9	23,940	1,854
Combo Card; Ethernet SCSI	FR-PCTBZ-AA	1	504	1	504	76
Ultra 68VHD 10M Cable Assembly	BN37A-10	1	175	34	5,950	0
D Shf 180W 1 Doc BLW Metric Blue	DS-BA356-JD	1	990	34	33,660	7,004
4/8GB 4MM SCSI DAT Tape Drive	TLZ09-VA	1	882	1	882	1,147
VT510,WHITE,NORTH AMER,NO KEY	VT510-AA	1	277	1	277	0
US/CANADA W95 KYBD WHIT	PCXLA-NA	1	18	1	18	0
9.1GB 7200RPM UltraSCSI 16bit	DS-RZ1DA-VW	1	455	259	117,84	Spared
17-03212-05 8MP-8MP Patch Cable	BN25G-07	1	6	6	36	0
Linksys 8port 10baste T Switch	DEH3552	2	119	3	357	Spared
				Subtotal	**396,100**	**67,130**
Server Software						
5YR AS ES40 UNIX	FM-62UN9-	1	9,457	1		9,457
5YR As ES40 Unix	FM-62USM-	1	1,121	3		3,363
cVISN MC MG MUL SYS 1	QM-MQDAA-	1	697	1	697	
clearVISN MUL CDRM	QA-5FVAB-	1	69	1	69	
Dig UNIX Alpha	QA-MT4AA-	1	277	1	277	
Sybase Development Pricing (License)	Sybase	3	9,600	1	9,600	
Sybase Adaptive Server	Sybase	3	141,98	1	141,98	
1 Yr Sybase Maintenance	Sybase	3	25,663.	5		128,31
				Subtotal	**152,623**	**141,136**
Client Hardware						
Compaq ProLiant 1600	153552-001	1	3,036	4	12,144	1,976
Ethernet Dual Channel	338456-B21	1	329	8	2,632	Inc.
NC3132 Dual 10/100	338456-B22	1	263	8	2,104	Inc.
256MB SDRAM Dimm Memory	313616-B21	1	1,132	8	9,056	Inc.
18GB Drive	388144-B22	1	989	4	3,956	Inc.
V700 17" Color Monitor	325900-001	1	185	4	740	Inc.
				Subtotal	**30,632**	**1,976**
Client Software						
Application Optimizer WNT Server Edition	QB-641AA-SA	1	315	4	1,260	
Application Optimizer WNT Server	QT-641AA-XA	1	72	5		1,440
Sybase Open Client/DB Lib	Sybase	3	795	1	795	
				Subtotal	**2,055**	**1,440**
User Connectivity						
Compex Micro Hub/8-8 Port Hub	DEH2924	2	32.5	3481	113,13	Spared
				Subtotal	**113,133**	
Notes: ** 10% Spares**				**Total**	**$694,543**	**$211,682**

ICS Discount: 30% on Server Hardware/Software based on
ICS Discount: 10% on FM and FR numbers based on
Sybase Discount: 10%
1=IC System Solutions, 2=MicroWarehouse, 3=
Audited by InfoSizing

Five-Year Cost of Ownership: $906,225
tpmC Rating: 30,738.97
$ / tpmC: $29.48

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

Numeric Quantities Summary
Compaq AlphaServer ES40 4CPU C/S System

MQTH

MQTH, computed Maximum Qualified Throughput	30,738
% throughput difference, reported and reproducibility runs	.001

Response Times in seconds

Transaction	90th percentile	Average	Maximum
New-Order	1.538	0.755	15.603
Payment	1.669	0.822	13.673
Order-Status	1.698	0.895	12.644
Delivery (interactive)	0.420	0.313	9.961
Delivery (deferred)	2.403	1.240	17.395
Stock-Level	2.070	0.957	14.476
Menu	0.380	0.257	10.100

Transaction Mix in percent of total transactions

Transaction	Total Occurrences	Percentage
New-Order	3688676	44.980%
Payment	3526881	43.007%
Order-Status	328386	4.004%
Delivery (interactive)	328423	4.005%
Stock-Level	328392	4.004%

Keying/Think Times in seconds

Transaction	Minimum	Average	Maximum
New-Order	18.000/0.000	18.005/12.006	18.159/119.999
Payment	3.000/0.000	3.011/12.005	3.564/119.825
Order-Status	2.000/0.000	2.005/10.009	2.305/ 99.119
Delivery (interactive)	2.000/0.000	2.005/ 5.006	2.381/ 49.986
Stock-Level	2.000/0.000	2.005/ 5.006	2.381/ 49.843

Emulation Delay in seconds

Transaction	Response Time	Menu Response Time
New-Order	0.10	0.10
Payment	0.10	0.10
Order-Status	0.10	0.10
Delivery (interactive)	0.10	0.10
Stock-Level	0.10	0.10

Test Duration

Rampup up time	30 minutes
Measurement interval	120 minutes
Number of checkpoints	4
Number of New Order transactions	3688676
Number of transactions (all types) completed in measurement interval	8200758

Abstract

This report documents the compliance of Compaq Computer Corporation's and Sybase, Inc.'s TPC Benchmark C tests on the Compaq AlphaServer ES40 4 CPU client/server system with Version 3.4 of the TPC Benchmark C Standard Specification. Four Compaq ProLiant 1600s were used as the front-end clients. One of these systems was also used as a development system.

Two standard metrics, transactions-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 Information Paradigm is appended at the end of this report.

Table of Contents

Preface

This report documents the compliance of Compaq Computer Corporation and Sybase, Inc., and TPC Benchmark C (TPC-C) testing on the Compaq AlphaServer ES40 4 CPU client/server system with Version 3.4 of the *TPC Benchmark C Standard Specification*[1]. The TPC-C Standard represents an effort by Compaq Computer Corporation and other members of the Transaction Processing Performance Council (TPC) to create industry-wide benchmarks for evaluating the performance and price/performance of transaction processing systems.

These tests were run using the Application Optimizer WNT Server Edition transaction processing monitor, Sybase Adaptive Server Enterprise 11.9.3 relational database under the Compaq Tru64 UNIX V4.0G operating system. Four Compaq ProLiant 1600 computers were used as the front-end clients.

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

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

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

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 benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

---

[1] *TPC Benchmark C Standard Specification*, Transaction Processing Performance Council, Version 3.4

Document Structure

This *TPC Benchmark C Non 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 Remote Terminal Emulator (RTE) code used to generate and record transactions.
- Appendix D contains the tunable parameters.
- Appendix E contains the independent auditor's report on the compliance of this disclosure with the benchmark specifications.
- Appendix F contains third-party price quotations.

Additional Copies

To request additional copies of this report, please contact:

Administrator, TPC Benchmark Reports
HPS Benchmark Performance Engineering
Compaq Computer Corporation
110 Spit Brook Road
(ZK02-3/M31)
Nashua, NH 03062
U.S.A.

FAX number: 603-884-6082

TPC Benchmark C Non 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 type 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.

1. General Items

1.1 Order and Titles

The order and titles of sections in the Test Sponsor's Full Disclosure Report must correspond with the order and titles for the 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.

1.2 Summary Statement

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

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

1.3 Numerical Quantities Summary

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

- *measurement interval in minutes,*
- *number of checkpoints in the measurement interval,*
- *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.

1.4 Application Program

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

Appendix A contains the DEC C application code and the Sybase stored procedures.

1.5 Sponsor

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

This benchmark test was sponsored by Compaq Computer Corporation and Sybase, Inc., and attested to by Information Paradigm.

1.6 Parameters and Options

Settings 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 D contains the tunable parameters used in the TPC-C tests.

1.7 Configuration Diagrams

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

- *Number and type of processors*
- *Size of allocated memory, and any specific mapping/partitioning 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.)*

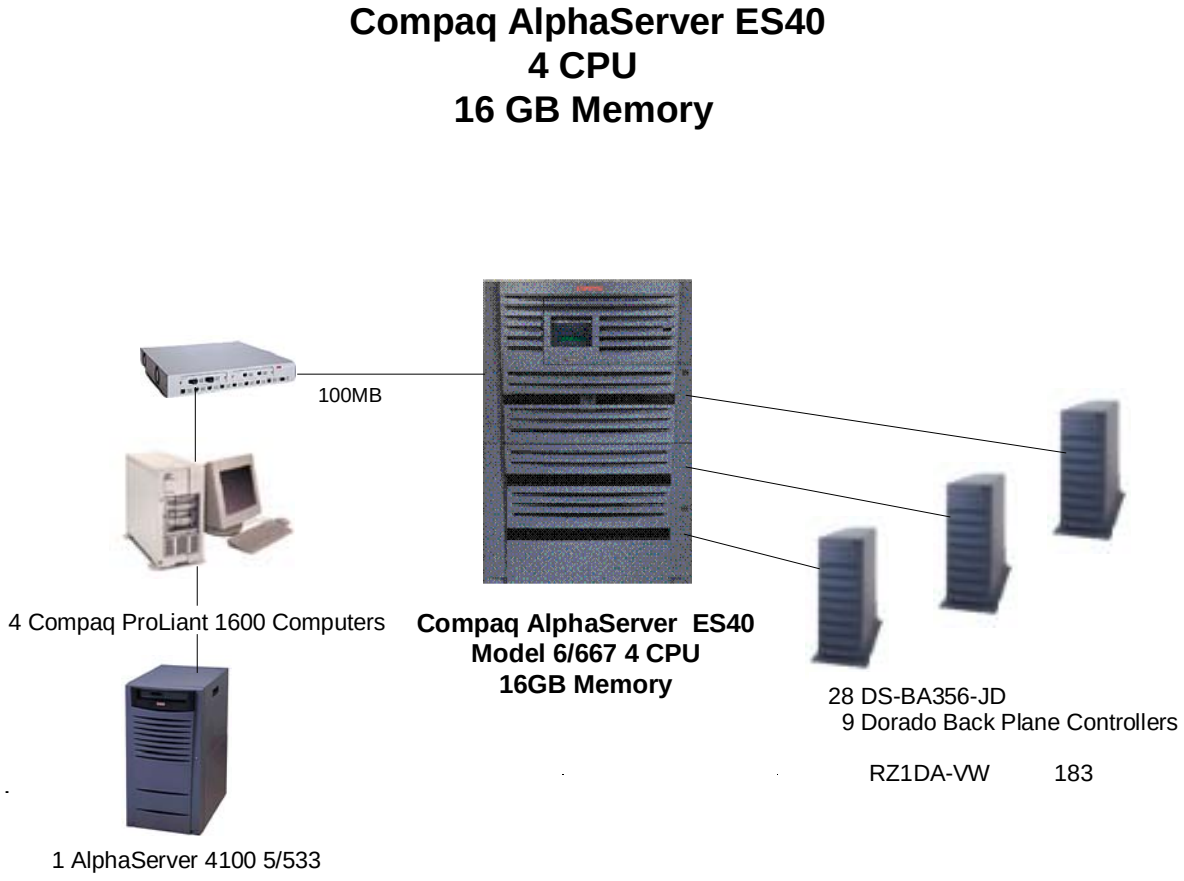
The TPC-C terminal users were emulated by Compaq's Portable Remote Terminal Emulator (PRTE) software, which ran on one AlphaServer 4100 5/533. The emulated terminal was connected using HTTP (HyperText Transport Protocol) to a client node.

The clients consisted of four Compaq ProLiant 1600 computers running the TPC-C web client screen and application software. Each emulated terminal ran the TPC-C client application and made a request to the Sybase Adaptive Server using SQL Server dblib calls.

The server was a Compaq AlphaServer ES40 4 CPU system running Compaq Tru64 UNIX V4.0G. The measured server was configured with 183 RZ1DA-VW. The priced configuration consists of 259 RZ1DA-VW (9.1GB disks, which includes an additional 10% spareable disks).

Measured Configuration

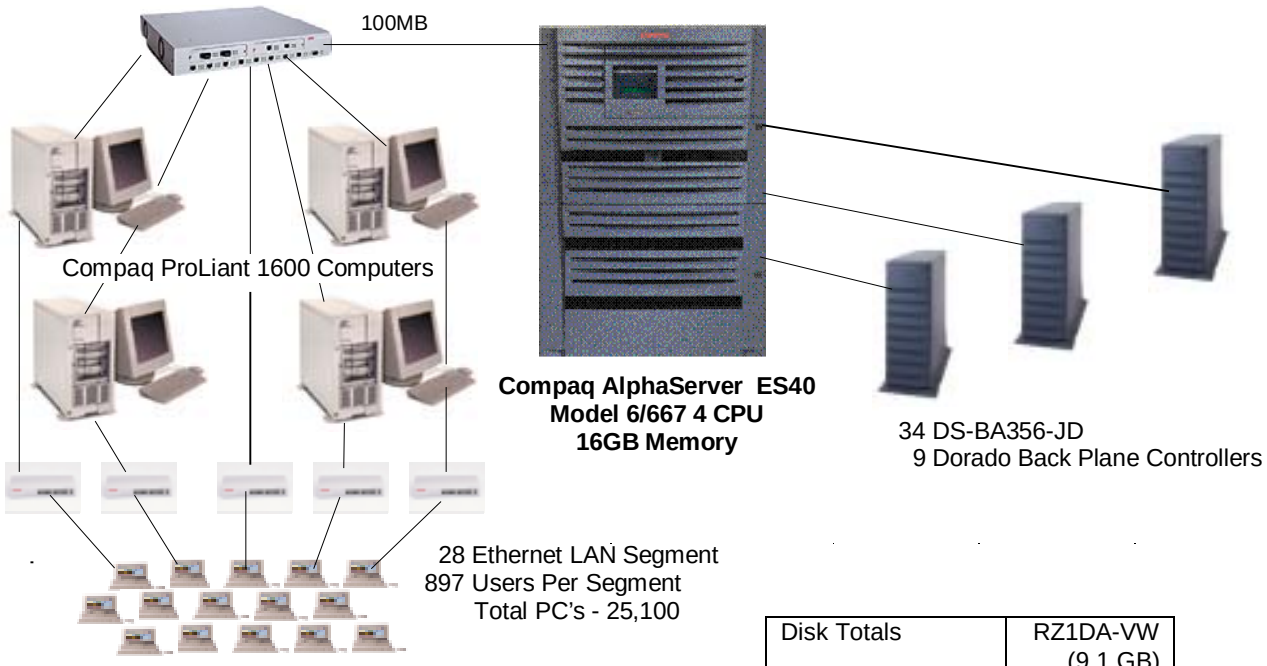
The following figure represents the measured configuration. The benchmark system used a remote terminal emulator (RTE) to initiate transactions and measure response times of transactions, as well as record various data for each transaction.



Priced System Configuration

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

Compaq AlphaServer ES40
4CPU
16 GB Memory



Disk Totals	RZ1DA-VW (9.1 GB)
DB + system disks	235
10% Spare disks	24
Total	259

2. Logical Database Design Related Items

2.1 Table Definitions

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

Appendix B contains the database definition files that were used to set up the database.

2.2 Table Organization

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

Physical space was allocated to Sybase Adaptive Server on the server disks according to the details provided in Appendix B. The charts in Section 5.2 also describe the layout. The size of the segments on each disk was calculated to provide even distribution of data across the disk subsystem. A fill factor was used on the Warehouse and District tables to minimize the amount of data per page. The indices were defined at table definition and were built at the initial table load by executing the database build script in Appendix B.

2.3 Insert/Delete Operations

It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C 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 maximum key value for these new rows.

There were no restrictions on insert and/or delete operations to any of the tables. The space required for an additional 5**%** of the initial table cardinality was allocated to Sybase Adaptive Server and priced as static space.

2.4 Disclosure of Partitioning

While there are 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.

Horizontal partitioning was used on the HISTORY table using functionality provided by Sybase Adaptive Server.

2.5 Replication of Tables

Replication of tables, if used, must be disclosed.

No tables were replicated in this benchmark test.

2.6 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 attributes were added or replicated in this benchmark test.

3. Transaction and Terminal Profiles Related Items

3.1 Random Number Generation

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

Random numbers were generated using the drand48() and lrand48() UNIX calls. These functions generate pseudo random numbers using the linear congruential algorithm and 48-bit integer arithmetic. The random number generators are initially seeded using the srand48() call.

3.2 Terminal Input/Output Screen Layouts

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

The screen layouts match the *TPC Benchmark C Standard Specification*.

3.3 Features in Priced Terminals

The method used to verify that the priced terminals provide all the features described in Clause 2.2.2.4 must be explained.

Each of the five transaction types was tested by the auditor from an Intel 486DX2/66 running Windows NT 4.0 SP2 and Netscape Navigator V3.0. The auditor verified that all the features specified in Clause 2.2.2.4 were provided. Any PC configured with any WWW browser will properly display the TPC-C screens.

3.4 Presentation Managers

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

The code to generate the input and menu screens and display the results runs on the front-end clients. The data is passed to the user's PC using the HTML (HyperText Markup Language) format, which can be displayed with any Web browser, such as Netscape Navigator or MicroSoft Internet Explorer. Both of these products come as standard software with many operating systems.

3.5 Home and Remote Order-Lines

The percentage of home and remote order-lines in the New-Order transactions must be provided.

The table in Section 3.10 shows the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

3.6 New Order Roll Back Transition

The percentage of New-Order transactions that were rolled back as a result of an invalid item number musts be provided.

The table in Section 3.10 shows the percentage of New-Order transactions that were rolled back due to an invalid item being entered.

3.7 Number of Order-Lines

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

The table in Section 3.10 shows the average number of items ordered per New-Order transaction.

3.8 Home and Remote Payment Transactions

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

The table in Section 3.10 shows the percentage of home and remote transactions that occurred during the measurement period for the Payment transactions.

3.9 Non-Primary Key Access in Payment and Order-Status

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

The table in Section 3.10 shows the percentage of non-primary key accesses to the database by the Payment and Order-Status transactions.

3.10 Skipped Deliveries

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

The following table summarizes the data required for disclosure from Sections 3.5 through 3.10. The range of acceptable and the measured results are listed.

Description	Section	Acceptible Requirement	Measured Result
% home order lines in New Order	3.5	(98.05 - 99.05)	99.00%
% remote order lines in New Order	3.5	(0.95 - 1.05)	1.00%
% New Order transactions rolled back	3.6	(0.9 - 1.1)	1.00%
Average number of items per order	3.7	(9.5 - 10.5)	10.00
% home Payment transactions	3.8	(84.0 - 86.0)	85.01%
% remote Payment transactions	3.8	(14.0 - 16.0)	14.99%
% non-primary key access, Payment	3.9	(57.0-63.0)	59.99%
% non-primary key access, Order Status	3.9	(57.0-63.0)	59.95%
% skipped deliveries	3.10	(0.0 - 1.0)	0%

3.11 Transaction Mix

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

The following table summarizes the transaction mix.

Transaction Mix (percent)

Transaction Type	New-Order	Payment	Order-Status	Delivery	Stock-Level
Min. Requirement	NA	43.000%	4.000%	4.000%	4.000%
Measured Result	44.980%	43.007%	4.004%	4.005%	4.004%

3.12 Queuing Mechanism

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

The delivery transactions were transmitted a separate delisrv process using a pipe. The pipe acts as a FIFO queue. The delisrv has multiple threads that can do the delivery transaction in the database. The delivery data is written to a file. The number of threads in the delisrv is configurable.

4. Transaction and System Properties Related Items

4.1 ACID Properties

The results of the ACID test 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.

Clause 3 of the TPC Benchmark C Standard Specification lists specific tests to ensure the Atomicity, Consistency, Isolation, and Durability (ACID) properties of the SUT. The following subsections show how the tests required in Clause 3 were performed and the results verified. All mechanisms needed to ensure full ACID properties were enabled during both the measurement and test periods. A fully-sized, 2510 warehouse database was used for the failure of memory and interruption, and consistency tests. Atomicity and Isolation with remaining durability were performed on a previous audit.

4.2 Atomicity Tests

The system under test must guarantee that 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.

This test was previously performed and waived by the auditor.

4.3 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.

The consistency of the database was checked before and after the power fail Durability test. The following consistency conditions were run and verified that all four conditions were met:

4.3.1 Consistency Condition 1

Entries in the WAREHOUSE and DISTRICT tables satisfy:

W_YTD = sum(D_YTD) for each warehouse defined by (W_ID = D_W_ID)

4.3.2 Consistency Condition 2

Entries in the DISTRICT, ORDER, and NEW-ORDER tables satisfy:

D_NEXT_O_ID - 1 = max(O_ID) = max(NO_O_ID) for each district defined by

(D_W_ID = O_W_ID = NO_W_ID) and (D_ID = O_D_ID = NO_D_ID)

4.3.3 Consistency Condition 3

Entries in the NEW-ORDER table satisfy:

max(NO_O_ID) - min(NO_O_ID) = [# of rows in NEW-ORDER of this district] for each district defined by NO_W_ID and NO_D_ID

4.3.4 Consistency Condition 4

Entries in the ORDER and ORDER-LINE tables satisfy:

sum(O_OL_CNT) = [# of rows in ORDER-LINE of this district] for each district defined by (O_W_ID = OL_W_ID) and (O_D_ID = OL_D_ID)

4.4 Isolation Tests

The TPC Benchmark C Standard Revision 3.3.2 defines seven required tests to be performed to demonstrate that the required levels of transaction isolation are met. All seven required tests were performed successfully. In addition to those seven tests, two more tests specified by the auditor were performed successfully. These additional tests demonstrated phantom protection within any mix of TPC-C transactions.

This test was previously performed and waived by the auditor.

4.5 Durability Tests

The tested system must guarantee the ability to preserve the effects of committed transactions and ensure 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).*

4.5.1 Failure of Durable Medium containing TPC-C Database Tables and Failure of Durable Medium containing Recovery Log Data

This test was previously performed and waived by the auditor.

4.5.2 Failure of Memory and Instantaneous Interruption

This test was conducted on the fully scaled 2510 warehouse database using 25,100 emulated PCs.

1. The current number of orders in the database was counted, giving ORDER_COUNT_BEFORE.

   1a. Consistency was verified.
2. A test was started and allowed to run at steady state for 5 minutes.
3. The system was powered off.
4. The test was aborted on the RTE.
5. The system was powered back on and rebooted.
6. Sybase Adaptive Server was restarted and recovered the database from the transaction log.

7. The current number of orders in the database was counted giving ORDER_COUNT_AFTER. It was verified that ORDER_COUNT_AFTER - ORDER_COUNT_BEFORE was greater than or equal to the number of committed orders recorded by the PRTE.
8. Several orders recorded by the PRTE were checked in the database to make sure they existed.
9. Consistency was verified again.

5. Scaling and Database Population Related Items

5.1 Cardinalities of the Database Tables

The cardinality (e.g. the number of rows) of each table, as it existed at the start of the benchmark run (see Clause 4.2), must be disclosed. If the database was over-scaled and inactive rows of the WAREHOUSE table were deleted (see Clause 4.2.2), the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed.

The initial cardinalities of the database tables are shown in the following table.

Table Name	Cardinality
WAREHOUSE	2510
DISTRICT	26000
CUSTOMER	78000000
HISTORY	78000000
NEW-ORDER	23400000
ORDER	78000000
ORDERLINE	780000000
STOCK	260000000
ITEM	100000

5.2 Distribution of Tables and Logs

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

The following benchmark configuration table indicates how the database files were allocated on the tested system to meet the 8-hour steady state requirement.

The distribution of the database tables over all the disks of the priced system is an extension of the distribution in the tested system. One hundred eighty (180) day storage requirements are satisfied with the unused space on the tested system disks. System was configured with enough mirrored log disk capacity to support 8 hours of run time.

Distribution of Data on the AlphaServer ES40

/dev/rri513b	log01	log	100.00%
/dev/rri1024h	cidx01	customer index	50.00%
/dev/rri384h	cidx02	customer index	50.00%
/dev/rri0f	cust01	customer	6.60%
/dev/rri128f	cust02	customer	6.60%
/dev/rri256f	cust03	customer	6.60%
/dev/rri384f	cust04	customer	6.60%
/dev/rri640f	cust05	customer	6.60%
/dev/rri768f	cust06	customer	6.60%
/dev/rri896e	cust07	customer	6.60%
/dev/rri1024e	cust08	customer	6.60%
/dev/rri0g	cust09	customer	6.60%

/dev/rri640g	cust10	customer	6.60%
/dev/rri768g	cust11	customer	6.60%
/dev/rri896f	cust12	customer	6.60%
/dev/rri1024f	cust13	customer	6.60%
/dev/rri896g	cust14	customer	6.60%
/dev/rri1024g	cust15	customer	6.60%
/dev/rri256g	history01	history	100.00%
/dev/rri128g	Master	master	100.00%
/dev/rri0h	orders01	orders	33.30%
/dev/rri768h	orders02	orders	33.30%
/dev/rri896h	orders03	orders	33.30%
/dev/rri384g	ordlne01	order_line	25.00%
/dev/rri640h	ordlne02	order_line	25.00%
/dev/rri256h	ordlne03	order_line	25.00%
/dev/rri128h	ordlne04	order_line	25.00%
/dev/rri0a	stock01	stock	3.30%
/dev/rri128a	stock02	stock	3.30%
/dev/rri256a	stock03	stock	3.30%
/dev/rri384a	stock04	stock	3.30%
/dev/rri640a	stock05	stock	3.30%
/dev/rri768a	stock06	stock	3.30%
/dev/rri896a	stock07	stock	3.30%
/dev/rri1024a	stock08	stock	3.30%
/dev/rri0b	stock09	stock	3.30%
/dev/rri128b	stock10	stock	3.30%
/dev/rri256b	stock11	stock	3.30%
/dev/rri384b	stock12	stock	3.30%
/dev/rri640b	stock13	stock	3.30%
/dev/rri768b	stock14	stock	3.30%
/dev/rri896b	stock15	stock	3.30%
/dev/rri1024b	stock16	stock	3.30%
/dev/rri0d	stock17	stock	3.30%
/dev/rri128d	stock18	stock	3.30%
/dev/rri256d	stock19	stock	3.30%
/dev/rri384d	stock20	stock	3.30%
/dev/rri640d	stock21	stock	3.30%
/dev/rri768d	stock22	stock	3.30%
/dev/rri896d	stock23	stock	3.30%
/dev/rri1024d	stock24	stock	3.30%
/dev/rri0e	stock25	stock	3.30%
/dev/rri128e	stock26	stock	3.30%
/dev/rri256e	stock27	stock	3.30%
/dev/rri384e	stock28	stock	3.30%
/dev/rri640e	stock29	stock	3.30%
/dev/rri768e	stock30	stock	3.30%

5.3 Type of Database

A statement must be provided that describes:

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

The database used for this testing was Sybase Adaptive Server Enterprise 11.9.3 from Sybase Incorporated. Sybase Adaptive Server is a relational DBMS.

The database was built for 2600 warehouses.

5.4 Database Partitions/Replications

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

Horizontal partitioning was used on the History Table. The functionality for this was provided by Sybase Adaptive Server. For further details of the partitioning of the database, see Appendix B.

5.5 180-Days Space Requirement

The calculations for arriving at the 180-day space computations, as defined in Clause 4.2.3 must be disclosed.

Note : Numbers are in KBytes unless otherwise specified

Warehouses	2510	**tpmC**	30738.967	**tpmC/W**	12.25

Table	**Rows**	**Data**	**Index**	**5% Space**	**8H Space**	**Total Space**
Warehouse	2,600	5,200	22	261		5,483
District	26,000	52,000	210	2,611		54,821
Item	100,000	9,524	86	192		9,802
New-order	23,400,000	255,738	3,082		52,000	310,820
History	78,000,000	4,368,032	0		686,023	5,054,055
Orders	78,000,000	2,108,110	25,396		335,079	2,468,585
Customer	78,000,000	52,000,000	4,349,884	1.126.998		57,476,882
Order-line	780,000,000	47,272,728	622,006		7,522,130	55,416,864
Stock	260,000,000	86,800,000	479,564	1.745.591		89,025,155
Totals		192,871,332	5,480,250	2.875.653	8,595,232	209,822,467

Segment	**LogDev Cnt.**	**Seg. Size**	**Needed**	**Overhead**	**Not Needed**
wdino	1	1,331,200	384,735	3,847	942,618
history	1	6,656,000	5,104,596	51,046	1,500,358
order	3	3,326,976	2,493,271	24,933	808,773
customer	15	58,941,440	58,051,650	580,517	309,273
order_line	4	66,560,000	55,971,033	559,710	10,029,257
stock	30	92,098,560	89,915,407	899,154	1,283,999
Totals		228,914,176	211,920,692	2.119.207	14,874,277

Dynamic space	52,297,651	Sum of Data for Order, Order-Line and History (excluding free extents)
Static space	151,048,791	Data + Index + 5% Space + Overhead - Dynamic space
Free space	10,693,457	Total Seg. Size - Dynamic Space - Static Space - Not Needed
Daily growth	9,892,774	(Dynamic space/W * 62.5)* tpmC
Daily spread	(4,145,704)	Free space - 1.5 * Daily growth (zero if negative)
180 day (KB)	1,931,748,08	Static space + 180 (daily growth + daily spread)
180 day (GB)	1842.26	Excludes OS, Paging and RDBMS Logs
Log per N-O txn	1.9	Number of 2K blocks per New-Order transaction
8 Hour Log	53.47	

Total Space (GB) Required on System (except for logs)

180 day (GB)	1,842.26
	1,842.26

Space Allocated for Benchmarked

(Excluding Drive Type	Number	Cap.	Total Capacity (GB)	Needed Space	Space Need
DS-RZ1DA-VW	168	8.47	1423.46	419	50
			1423.46		

Space Allocated for Logs				
Drive Type	Number	Cap. GB)	Capacity (GB)	
DS-RZ1DA-VW	14	8.48	118.65	unmirrored capacity
Space Needed for Logs				
DS-RZ1DA-VW	7	8.48	59.33	unmirrored capacity
Total	14			mirrored devices

Total Priced devices		
System Disk	1	
For data	232	mirrored logs + space allocated
Spares	24	10% extra

6. Performance Metrics and Response Time Related Items

6.1 Reporting All Data

Measured tpmC must be reported.

All the data required by Clause 5 is reported below in Section 6.2 through 6.10. The measured tpmC for the Compaq AlphaServer ES40 4 CPU C/S configuration was 30,738 tpmC.

6.2 Response Times

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

Response Times in seconds

Transaction	90th percentile	Average	Maximum
New-Order	1.538	0.755	15.603
Payment	1.669	0.822	13.673
Order-Status	1.698	0.895	12.644
Delivery (interactive)	0.420	0.313	9.961
Delivery (deferred)	2.403	1.240	17.395
Stock-Level	2.070	0.957	14.476
Menu	0.380	0.257	10.100

6.3 Think and Keying Times

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

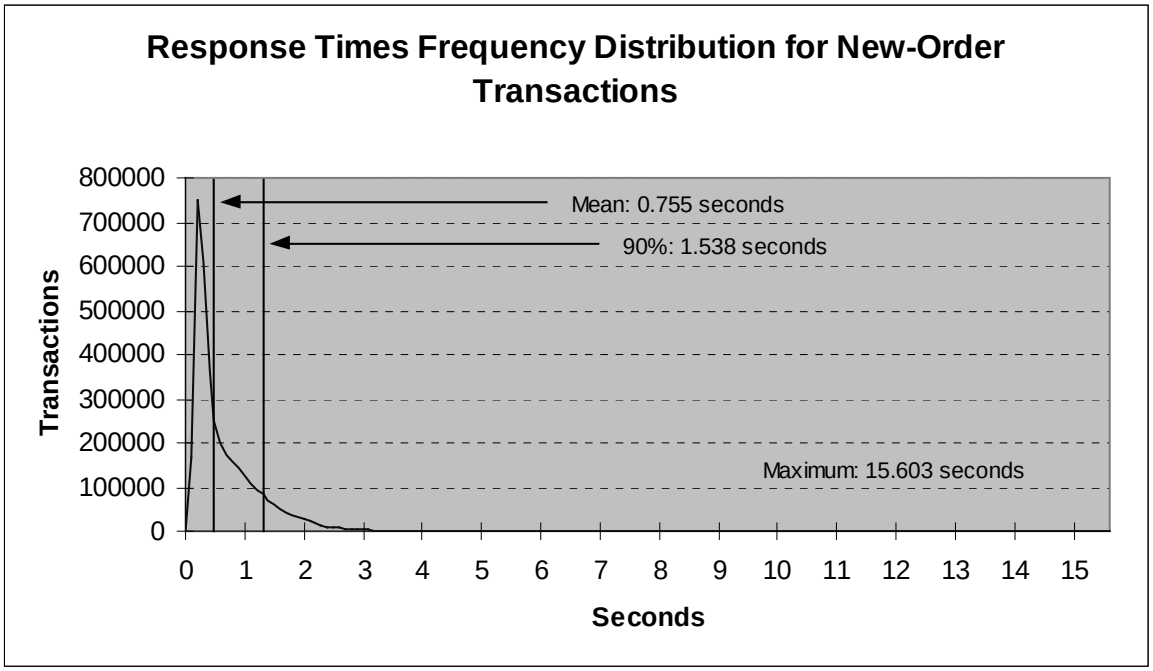
Keying/Think Times in seconds

Transaction	Minimum	Average	Maximum
New-Order	18.000/0.000	18.005/12.006	18.159/119.999
Payment	3.000/0.000	3.011/12.005	3.564/119.825
Order-Status	2.000/0.000	2.005/10.009	2.305/ 99.119
Delivery (interactive)	2.000/0.000	2.005/ 5.006	2.381/ 49.986
Stock-Level	2.000/0.000	2.005/ 5.006	2.381/ 49.843

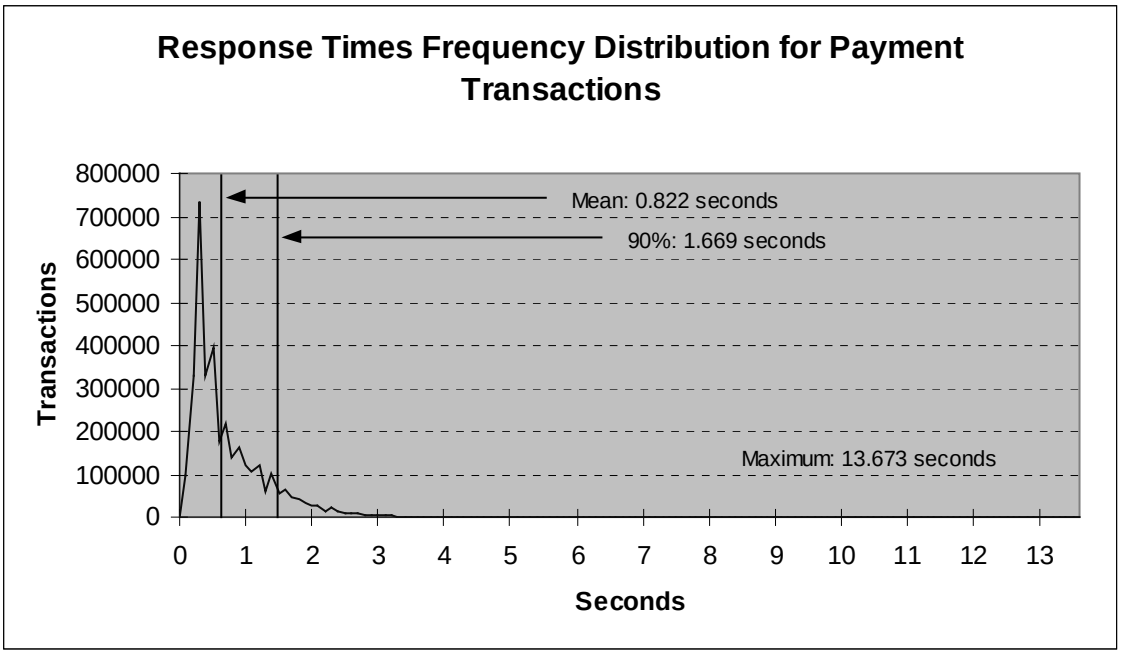
6.4 Response Times Frequency Distribution

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

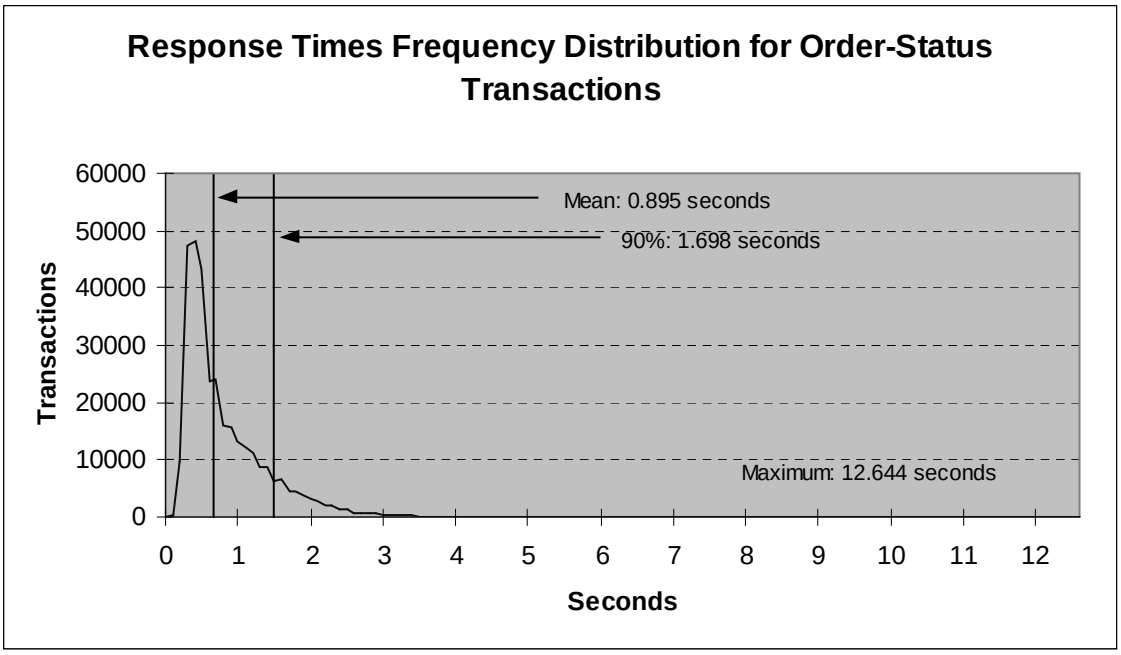
Response Times Frequency Distribution for New-Order Transactions



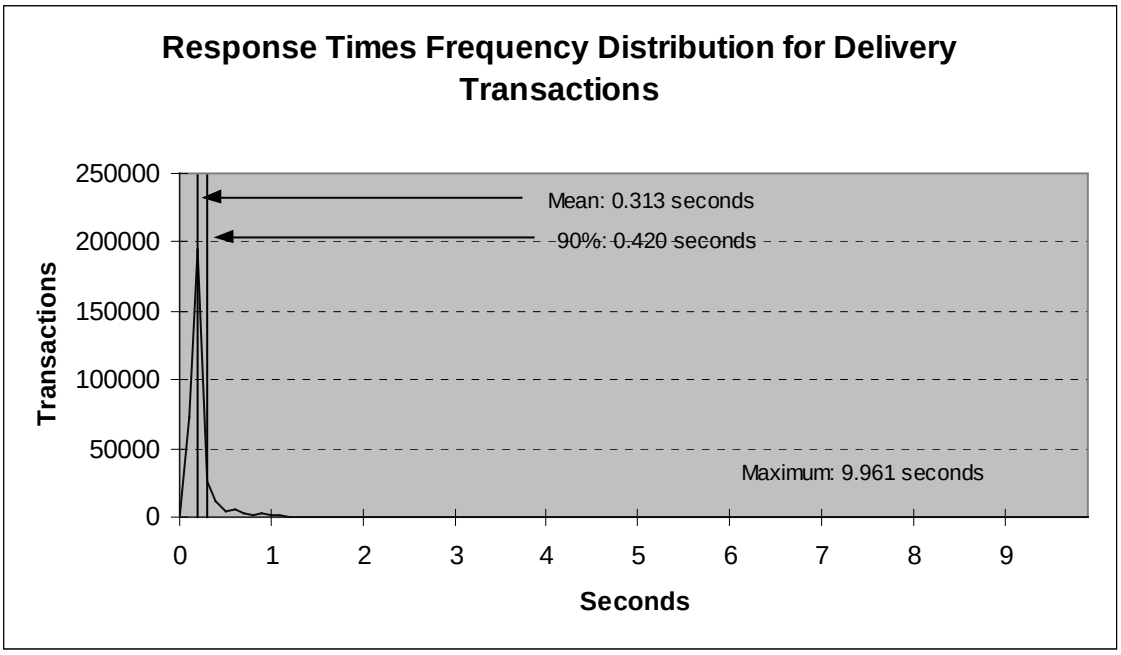
Response Times Frequency Distribution for Payment Transactions



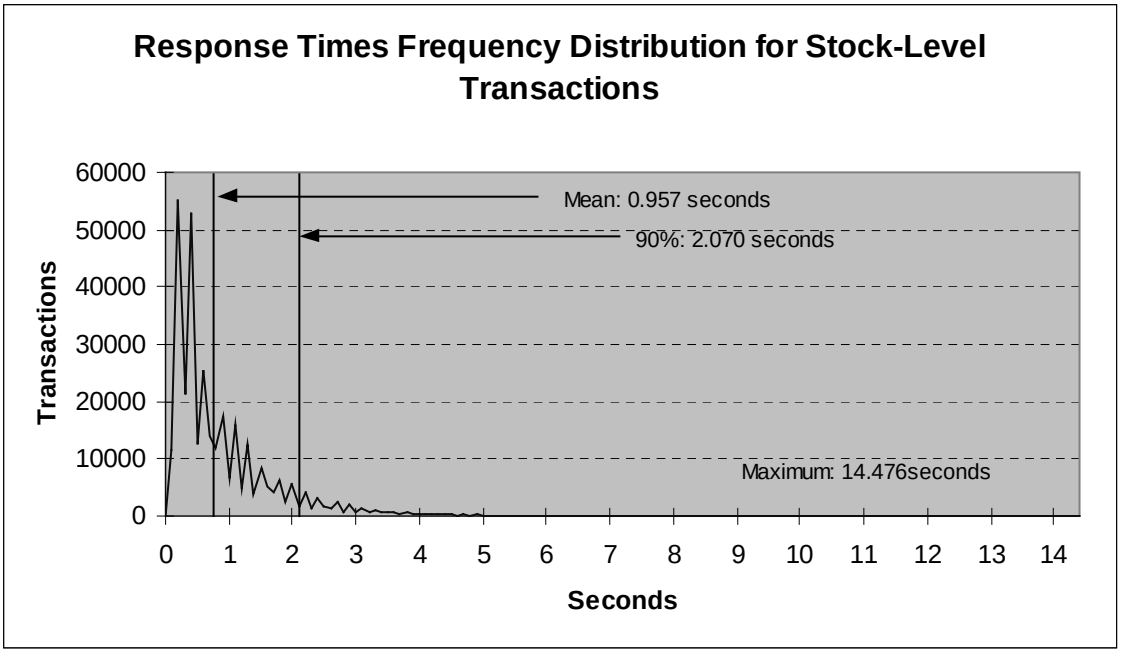
Response Times Frequency Distribution for Order-Status Transactions



Response Times Frequency Distribution for Delivery Transactions



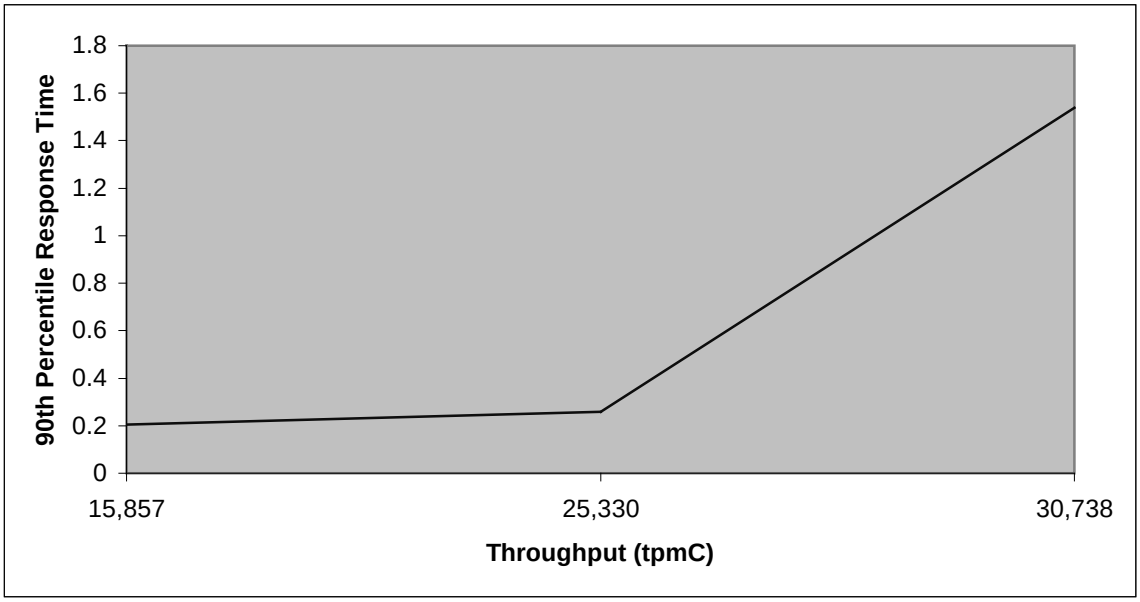
Response Times Frequency Distribution for Stock-Level Transactions



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.

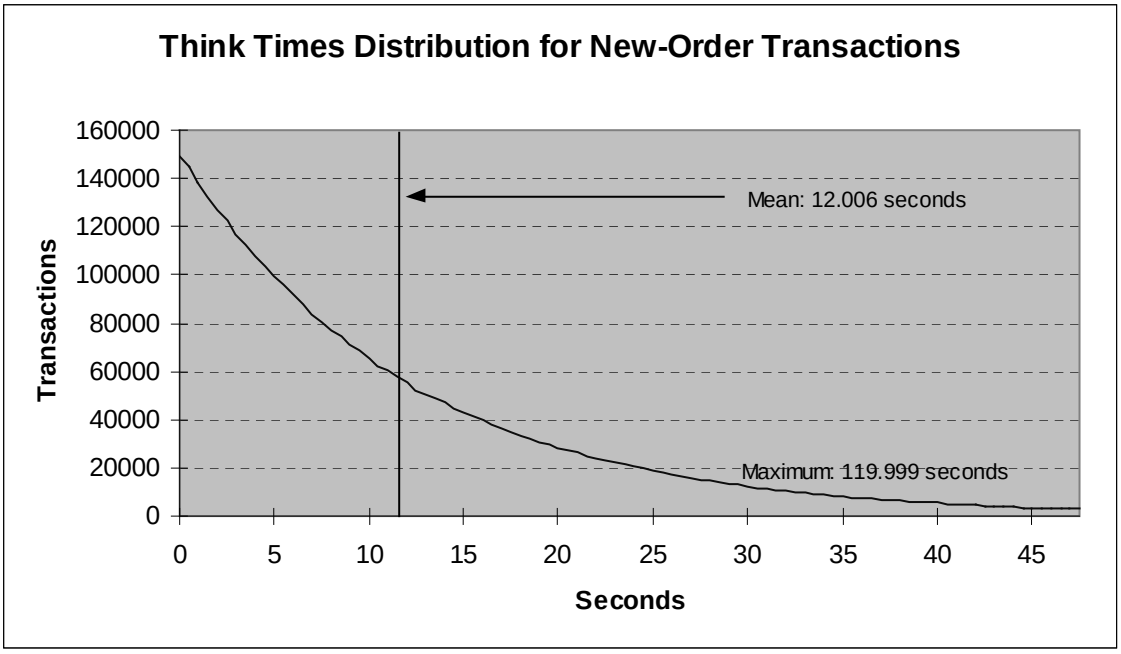
New-Order Response Time versus Throughput



Think Times Frequency Distribution

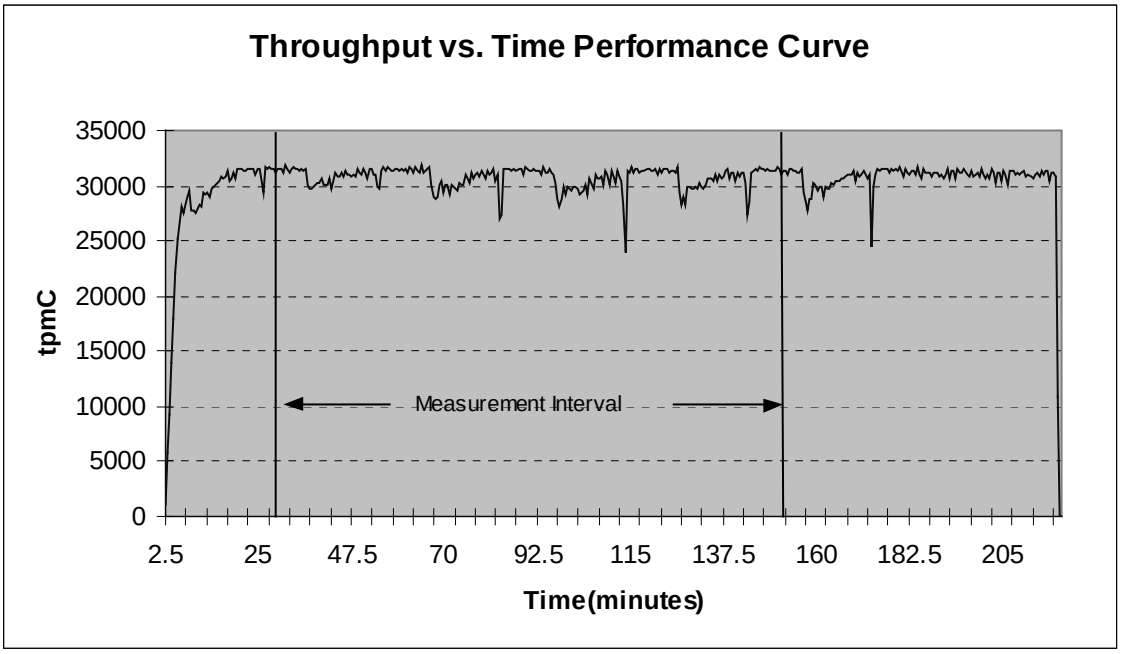
Think times frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type.

Think Times Distribution for New-Order Transactions



6.5 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.



6.6 Steady State

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

Confirmation that the SUT has reached steady state prior to the beginning of the data collection measurement interval is based on a visual inspection of the plots of tpmC versus time.

The graph in Section 6.6 plots the average tpmC versus time, averaged over 30 second intervals, and shows that steady state was reached before the data was collected. The ramp-up and steady-state stages are clearly visible.

6.6.1 Transaction Flow

For each of the TPC Benchmark C transaction types, the following steps are executed.

Compaq Application Optimizer WNT Server Edition was used as the transaction manager (TM). Each transaction was divided into two pieces. The front-end portion handled all screen I/O, while the back-end portion handled all database operations. Both front-end and back-end pieces ran on the client system.

The front-end portion communicates with the back-end portion through shared memory. The back-end portion communicates with the server system over ethernet using Sybase Adaptive DB-Library/C calls. Compaq Application Optimizer WNT Server Edition routes the transaction and balances the load according to values in the registry governing the number of processing procedure servers. The transaction flow is described below.

- Each TPC-C user invokes the TPC-C main front-end portion.
- The front-end portion displays the TPC-C transaction menu on the user terminal.
- The TPC-C user chooses the transaction type and proceeds to fill the screen fields required for that transaction.
- The TPC-C front-end accepts all values entered by the user and transmits those values to one of the TPC-C back-end processing procedures. The back-end procedures are generic in that they can serve any of the transaction types.
- A TPC-C back-end processing procedure receives a workspace and proceeds to execute all database operations related to the transaction type specified. All the information entered on the user terminal is contained in the Compaq Application Optimizer WNT Server Edition workspace.
- Once the transaction is committed, the TPC-C back-end processing procedure loads the workspace with the transaction output and returns control to the Compaq Application Optimizer WNT Server Edition transaction manager.
- Compaq Application Optimizer WNT Server Edition passes the workspace back to the TPC-C main front-end.

6.7 Database Transaction

All database operations are performed by the TPC-C back-end programs. The process is described next.

- Using Sybase Open Client DB-Library calls, the TPC-C back-end program interacts with Sybase Adaptive Server to perform SQL data manipulations such as update, select, delete, and insert, as required by the transaction. After all database operations are performed for a transaction, the transaction is committed.
- Sybase Adaptive Server proceeds to update the database as follows:

When Sybase Adaptive Server changes a database table with an update, insert, or delete operation, the change is initially made in memory, not on disk. When there is not enough space in the memory buffer to read in or write additional data pages, Sybase Adaptive Server will make space by flushing some modified pages to disk. Modified pages are also written to disk when a checkpoint occurs. Before a change is made to the database, it is first recorded in the transaction log. This ensures that the database can be recovered completely in the event of a failure. Using the transaction log, transactions that started but did not complete prior to a failure can be undone, and transactions recorded as complete in the transaction log but not yet written to disk can be redone.

6.8 Work Performed During Steady State

A description of how the work normally performed during a sustained test actually occurred during the measurement interval must be reported.

For each of the TPC Benchmark C transaction types, the following steps are executed.

Each emulated user starts an Internet browser and asks to attach to the tpcc.dll on the desired client. The tpcc.dll formats the menus, input forms and data output using HTML (HyperText Markup Language). The HTML strings are transmitted over TCP/IP back to the client, where they can be displayed by any Web Browser software. The tpcc.dll on the client is run under the control of the Internet Information Server.

Deferred delivery processing is handled by a separate process, delisrv. The tpcc.dll takes delivery requests from the user and puts them on a pipe for the delisrv to handle.

Transactions are submitted by the RTE in accordance with the rules of the TPC-C benchmark. The emulated user chooses a transaction from the menu. The RTE records the time it takes from selecting the menu item to receiving the requested form. Data is generated for input to the form,

then the user waits the specified keying time. The submit is sent and the RTE records the time it takes for the transaction to be processed and all the output data to be returned. The user then waits for the randomly generated think time before starting the process over again. All timings taken by the RTE generate a start and end timestamp. Keying and think times are calculated as the difference between end-time of a timing to the start of the next.

The database records transactions in the database tables and also the transaction log. Writes to the database may stay in SQL Server's in-memory data cache for a while before being written to disk. Checkpoints are done every 30 minutes to flush the contents of the data cache to disk.

6.9 Determining Reproducibility

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

The experiment at the maximum targeted tpmC level was repeated once to ensure reproducibility. The computed tpmC for each experiment was within .001% of the reported tpmC, and all 90th percentile response times were under their respective limits.

6.10 Duration of Measurement Period

A statement of the duration of the measurement period for the reported maximum qualified throughput (tpmC) must be included.

Each experiment was run for a minimum of 2 hours. The data collection period was 2 hours and started 30 minutes after all simulated users had begun executing transactions followed by a 5 minute ramp-down.

6.11 Method of Regulation of the Transaction Mix

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

The weighted distribution method was used. No adjustments were made by the RTE scripts. See Appendix C for more details.

6.12 Percentage of the Total Mix

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 numbers must be disclosed. The average number of order-lines entered per New-Order transaction must be disclosed. The percentage of remote Payment transactions must be disclosed. The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed. The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed.

See Table in Section 3.10.

6.13 Percentage of New-Order Transactions Rolled Back

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

See Section 3.10.

6.14 Average Number of Order-Lines

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

See Section 3.10.

6.15 Percentage of Remote Order-Lines

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

See Section 3.10.

6.16 Percentage of Remote Payment Transactions

The percentage of remote Payment transactions must be disclosed.

See Section 3.10.

6.17 Percentage of Customer Selections

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

See Section 3.10.

6.18 Percentage of Delivery Transactions

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

See Section 3.10.

6.19 Number of Checkpoints

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.

A checkpoint is the process of writing all modified data pages to disk. The TPC-C benchmark on the Compaq AlphaServer ES40 4 CPU C/S system was set up to automatically checkpoint every 30 minutes. One checkpoint occurs during the warm-up period, and four occured during the measurement period.

7. SUT, Driver, and Communication Definition Related Items

7.1 Description of RTE

If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs (e.g., scripts) to the RTE had been used.

In order to simulate terminal users and record response times, a proprietary emulation package, PRTE, was used to simulate terminal users, generate random data, and record response times. This package runs on a processor that is distinct from the system under test. The scripts used to execute the tests are included as Appendix C.

7.2 Driver Functionality and Performance

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

Due to the large number of PCs and associated hardware that would be required to run these tests, a Remote Terminal Emulator was used to emulate the connected PCs and LAN.

As configured for this test, the driver software emulates the traffic that would be observed from the users' PCs connected by Ethernet to the DEChubs using HTTP (HyperText Transfer Protocol) over TCP/IP.

7.3 Functional Diagrams and Details 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 software and hardware functionality being performed on the Driver System, and its interface to the SUT must be disclosed (see Clause 6.6.3.6).

The diagrams in Section 1.7 show the tested and priced benchmark configurations.

7.4 Network Configurations and Driver System

The network configurations of both the tested service and the 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 (see Clause 6.6.4).

Section 1.7 in this report has a picture of the network configurations of both the tested service and the proposed (target) services.

In the priced configuration, the client machines are connected over 100 megabits per second (Mbps) ethernet. The emulated PCs are connected on 100 mbps LANs to the clients. Each client has 7 Ethernet ports, each of which is connected to a PC Lan with 897 PCs.

7.5 Network Bandwidth

The bandwidth of the network(s) 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 systems complies with the IEEE 802.3 standard and has a bandwidth of 100 megabits per second (Mbps).

The ethernet between the front-end clients and the server has a bandwidth of 100 Mbps.

7.6 Operator Intervention

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

No operator intervention was required.

8. Pricing Related Items

8.1 Hardware and Software Components

A detailed list of hardware and software used in the priced system. Each item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package pricing is used, contents of the package must be disclosed.

This section lists the separate components in the priced system.

8.1.1 Hardware Pricing

Compaq Computer Corporation's TPC Benchmark C tests used packaged hardware systems whenever possible to simplify configurations to the fewest number of line items.

All hardware and software products have been priced to satisfy the 5 year warranty and 5 x 8 with 4 hour response requirements.

The hardware prices for the Compaq AlphaServer ES40 4 CPU, 4 Compaq ProLiant 1600 computers and their associated components (e.g., memory, storage subsystem, etc.) are based on IC System Solutions, MicroWarehouse and Compaq Computer Corporation price quotations.

8.1.2 Software Pricing

The priced system uses the following software products:

- Compaq Tru64 UNIX operating system
- Compaq Application Optimizer WNT Server Edition
- Windows NT C++ compiler
- Sybase Adaptive Server relational database management system
- Microsoft NT Operating System

The license purchase includes 1 year of warranty service. Four years of additional software warranty is provided for a total of 5 years extended warranty. The software warranty and service level are the same as the service level for the hardware system on which the software operates.

The level of post-warranty software service is Layered Product Support (LPS).

8.1.3 Warranty Pricing

In addition to the base warranty, additional warranty has been priced to satisfy the 5-year TPC-C warranty requirements of all products.

8.1.4 Price Discounts

See Appendix F.

8.2 Availability Status

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

The version of Sybase Adaptive Server and Compaq Tru64 UNIX is presently available. The Compaq AlphaServer ES40 hardware components used in the measurement is available now. However, the DPT controllers will be available on March 17, 2000.

8.3 Performance and Price/Performance

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

The following table shows the measured tpmC and price/tpmC results for the tested systems:

CPU Model	Configuration	Software	tpmC	Price per tpmC $tpmC
Compaq AlphaServer ES40	4 CPU	Compaq Tru64 UNIX 4.0G Sybase Adaptive Server Enterprise 11.9.3	30,738	$29.48/tpmC

8.4 Country Specific Pricing

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

None.

8.5 Usage Pricing

For any usage pricing, the sponsor must disclose:

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

None.

9. Audit Related Items

9.1 Audit

If the bench benchmark has been independently audited, the auditor's name, address, phone number, and a brief audit summary report indicating compliance must be included in the Full Disclosure Report. A statement should be included, specifying when the complete audit report will become available and who to contact in order to obtain a copy.

Appendix E contains the complete independent auditor's letter by Francois Raab of Information Paradigm for the test described in this report.

Appendix A

Client Application

crestdl.c

```
*************************************************************************
*                                                                       *
* COPYRIGHT (c) 1998 BY                                                 *
* COMPAQ COMPUTER CORPORATION, *
* ALL RIGHTS RESERVED.                                                  *
*                                                                       *
* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  *
* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  *
* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  *
* TRANSFERRED.                                                          *
*                                                                       *
* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  *
* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY COMPAQ COMPUTER  *
* CORPORATION.                                                          *
*                                                                       *
* COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  *
* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                *
*                                                                       *
*                                                                       *
*************************************************************************

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


#include <tpcc_acmsxp.h>

#define MAX(a,b) ((a)>(b)?(a):(b))

int __cdecl
main(int argc, char *argv[])
{
 FILE *fptr;
 char filename[] = "tpcc_stdl_workspaces.stdl";
 char fname[132];
 char *dir;
 int  iMaxSize;

 dir = getenv( "USR_OBJ" );

 if( 1 < argc ) {
  if( 0 == strcmp( "-o", argv[1] ))
    dir = argv[2];
 }

 sprintf(fname, "%s\\%s", dir, filename);
 if ( (fptr=fopen(fname,"w")) == NULL)
 {
  printf("\nCould not open file %s\n",fname);
  exit (0);
 }

 iMaxSize = 0;
 iMaxSize = MAX(iMaxSize,sizeof(DeliveryData));
 iMaxSize = MAX(iMaxSize,sizeof(NewOrderData));
 iMaxSize = MAX(iMaxSize,sizeof(OrderStatusData));
 iMaxSize = MAX(iMaxSize,sizeof(PaymentData));
 iMaxSize = MAX(iMaxSize,sizeof(StockLevelData));
 fprintf(fptr, "RECORD io_login_wksp\n");
 fprintf(fptr, "\tlogin_data TEXT SIZE %d;\n",sizeof( LoginData ));
 fprintf(fptr, "END RECORD;\n\n");
 fprintf(fptr, "RECORD io_dy_wksp\n");
 fprintf(fptr, "\tdy_data TEXT SIZE %d;\n",sizeof( DeliveryData ));
 fprintf(fptr, "END RECORD;\n\n");
 fprintf(fptr, "RECORD io_no_wksp\n");
 fprintf(fptr, "\tno_data TEXT SIZE %d;\n",sizeof( NewOrderData ));
 fprintf(fptr, "END RECORD;\n\n");
 fprintf(fptr, "RECORD io_pt_wksp\n");
 fprintf(fptr, "\tpt_data TEXT SIZE %d;\n",sizeof( PaymentData ));
 fprintf(fptr, "END RECORD;\n\n");
 fprintf(fptr, "RECORD io_os_wksp\n");
 fprintf(fptr, "\tos_data TEXT SIZE %d;\n",sizeof( OrderStatusData ));
 fprintf(fptr, "END RECORD;\n\n");
 fprintf(fptr, "RECORD io_sl_wksp\n");
 fprintf(fptr, "\tsl_data TEXT SIZE %d;\n",sizeof( StockLevelData ));
 fprintf(fptr, "END RECORD;\n\n");
 fprintf(fptr, "RECORD io_gc_wksp\n");
 fprintf(fptr, "\tgc_data TEXT SIZE %d;\n", iMaxSize );
 fprintf(fptr, "END RECORD;\n\n");
 fprintf(fptr, "RECORD int_gc_wksp\n");
 fprintf(fptr, "\ttrans INTEGER;\n");
 fprintf(fptr, "END RECORD;\n\n");

 fclose(fptr);
 printf("\n File %s generated.\n",fname);

 return 0;
}
```

deli_cli.c

```
/*+*************************************************************************
 *                                                                       *
 * COPYRIGHT (c) 1997 BY                                                 *
 * COMPAQ COMPUTER CORPORATION, .                                        *
 * ALL RIGHTS RESERVED.                                                  *
 *                                                                       *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  *
 * TRANSFERRED.                                                          *
 *                                                                       *
 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  *
 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY COMPAQ COMPUTER  *
 * CORPORATION.                                                          *
 *                                                                       *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                *
 *                                                                       *
 *                                                                       *
 *************************************************************************

/*+
 * Abstract: This file contains functions for the delivery server client code.
 *
 * Author:  W Carr
 * Creation Date: July 1997
 *
 * Modified history:
 *
 *
*/

#define DELI_CLI_C

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

```
#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

#include <buf.h>
#include <deli_srv.h>



int DELIClientStartup( void )
{
 return ERR_SUCCESS;
}



int DELIClientShutdown( void )
{
 return ERR_SUCCESS;
}



int TPCCDelivery( pDeliveryData pDelivery,
                  pDeliveryData
CompletedDeliveries[DELIVERY_RESPONSE_COUNT] )
{
 size_t                  bw;
 size_t                  br;
 int                             ii;
 int                             status;


 if( 0 == pDelivery->queue_time )
  time( &pDelivery->queue_time );

 if( gbUsePipe ) {
  if( 0 == WriteFile( ghPipeInputWrite, pDelivery, sizeof(DeliveryDataInput),
                      &bw, NULL )) {
   status = GetLastError( );
   return ERR_DB_DELIVERY_NOT_QUEUED;
  }
 }
 else {



  pDelivery->delta_time = GetTickCount( );



  if( BUF_SUCCESS != bufwrite( &pDelivery, sizeof( pDeliveryData ),
                               &bw, INFINITE, inputbuf ))
   return ERR_DB_DELIVERY_NOT_QUEUED;

  for( ii = 0; ii < DELIVERY_RESPONSE_COUNT; ii++ ) {
   if( gbDisplayCompletions ) {
            status = bufread( &CompletedDeliveries[ii], sizeof( pDeliveryData ),
                              &br, 0, outputbuf );
            if( BUF_READTIMEOUT == status )

             CompletedDeliveries[ii] = NULL;
            else if( BUF_SUCCESS != status )
             return ERR_DELIVERY_OUTPUT_PIPE_READ;
   }
   else {

            CompletedDeliveries[ii] = NULL;
   }
  }
 }

 return ERR_DB_SUCCESS;
}
```

deli_cli.h

```
#ifndef DELI_CLI_H
#define DELI_CLI_H

/*+*****************************************************************************
 *                                                                             *
 * COPYRIGHT (c) 1997 BY                                                       *
 * COMPAQ COMPUTER CORPORATION, .                                              *
 * ALL RIGHTS RESERVED.                                                        *
 *                                                                             *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
 * ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
 * OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
 * TRANSFERRED.                                                            *
 *                                                                             *
 * THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
 * AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY COMPAQ COMPUTER  *
 * CORPORATION.                                                            *
 *                                                                             *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                  *
 *                                                                             *
 *                                                                             *
 *****************************************************************************/
/*+
 * Abstract: This file contains definitions and declarations for the
 *                              delivery server client code.
 *
 * Author: W Carr
 * Creation Date: July 1997
 *
 * Modified history:
 *
 *
*/

int DELIClientStartup( void );
int DELIClientShutdown( void );

#endif
```

deli_srv.c

```
/*+*****************************************************************************
 *                                                                             *
 * COPYRIGHT (c) 1997 BY                                                       *
 * COMPAQ COMPUTER CORPORATION, .                                              *
 * ALL RIGHTS RESERVED.                                                        *
 *                                                                             *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
 * ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
 * OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
 * TRANSFERRED.                                                            *
 *                                                                             *
 * THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
 * AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY COMPAQ COMPUTER  *
 * CORPORATION.                                                            *
 *                                                                             *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                  *
 *                                                                             *
 *                                                                             *
 *****************************************************************************/

/*+
```

```
 * Abstract: This file contains functions for the delivery server queue.
 *
 * Author:  W Carr
 * Creation Date: July 1997
 *
 * Modified history:
 *
 *
*/

#define DELI_SRV_C

#include <windows.h>
#include <winsock.h>
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#include <process.h>
#include <crtdbg.h>

#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

#include <buf.h>
#include <tpcc.h>
#include <deli_srv.h>

#define FILENAMESIZE 256

static int      giDeliInitStatus = ERR_SUCCESS;
static LONG             gDeliThreadStartCtr;
static HANDLE           gDeliThreadStartEvent;

static int      iNumThreads             = 4;
static int      iNumQueued              = 10;
static int      iDeadlockRetry          = 3;
static int      giLoginDelay            = 0;

static FILE   *fpLog = NULL;
static char    szLogPath[256] = {0};

static BOOL bDone;
static BOOL bLog = FALSE;
static BOOL bFlush = FALSE;
static BOOL bDirectConnect = FALSE;


static HANDLE           hPipeInputRead = INVALID_HANDLE_VALUE;


void                    DELIErrorMessage(int iError);
void                    DELILog( pDeliveryData pDelivery );
unsigned      __stdcall DELIThread( void *ptr );
int                     DELIReadRegistrySettings(void);



int
DELIGetTransportData( BOOL *dy_use_transport, int *num_dy_servers,
                                int *num_queued_deliveries, int
*num_queued_responses )
{
  int                                   status;

  if ( ERR_SUCCESS != (status = DELIReadRegistrySettings()) )
    return status;

  *dy_use_transport = !bDirectConnect;
  *num_dy_servers = iNumThreads;
  *num_queued_deliveries = iNumQueued;
  if( gbDisplayCompletions && !gbUsePipe )
    *num_queued_responses = iNumQueued;
  else
    *num_queued_responses = 0;

  return ERR_SUCCESS;
}



int
DELIServerStartup( BOOL dy_use_transport, int num_dy_servers,
                                int num_queued_deliveries, int
num_queued_responses )
{
  int                                   iError;
  int                                   ii;
  size_t                inputbufsize;
  size_t                outputbufsize;
  unsigned              tid;
  unsigned long                         ulhThread;
  HANDLE                hThread;

  if( gbUsePipe ) {

    inputbufsize = num_dy_servers *
      num_queued_deliveries * sizeof( DeliveryData );
    if(0 == CreatePipe(&hPipeInputRead, &ghPipeInputWrite, NULL,
inputbufsize)){
      iError = GetLastError( );
      return ERR_DELIVERY_PIPE_CREATE;
    }
  }
  else {

    inputbufsize = num_queued_deliveries * sizeof( pDeliveryData );
    if( BUF_SUCCESS != bufopen( inputbufsize, &inputbuf ))
      return ERR_DELIVERY_PIPE_OPEN;

    if( gbDisplayCompletions ) {

      outputbufsize = num_queued_responses * sizeof( pDeliveryData );
      if( BUF_SUCCESS != bufopen( outputbufsize, &outputbuf ))
                return ERR_DELIVERY_PIPE_OPEN;
    }
  }


  gDeliThreadStartCtr = iNumThreads;

  gDeliThreadStartEvent = CreateEvent( NULL, TRUE, FALSE, NULL );
  if ( gDeliThreadStartEvent == NULL )
    return ERR_CANT_CREATE_FORCE_THRED_STRT_EVENT;

  bDone = FALSE;

  if ( !bDone ){
    for( ii = 0; ii < iNumThreads; ii++ ) {
      ulhThread = _beginthreadex( NULL, 0, DELIThread, NULL, 0, &tid );
      if( 0 == ulhThread )
                return ERR_CANT_START_DELIVERY_THREAD;
      hThread = ( HANDLE )ulhThread;
      CloseHandle( hThread );
      if( !dy_use_transport )
                Sleep( giLoginDelay );
    }
  }

  WaitForSingleObject( gDeliThreadStartEvent, INFINITE );

  return giDeliInitStatus;
}
```

```
int
DELIServerShutdown( void )
{
 bDone = TRUE;

 if( gbUsePipe ) {
  CloseHandle( hPipeInputRead );
  CloseHandle( ghPipeInputWrite );
 }
 else {
  bufclose( inputbuf );
  bufclose( outputbuf );
 }

 if ( fpLog )
  fclose(fpLog);

 fpLog = NULL;

 return ERR_SUCCESS;
}



void
DELIErrorMessage(int iError)
{
 int ii;

 for( ii = 0; errorMsgs[ii].szMsg[0]; ii++ ) {
  if ( iError == errorMsgs[ii].iError ) {
   TPCCErr( "*Error(%d): %s\r\n", iError, errorMsgs[ii].szMsg );
   return;
  }
 }

 TPCCErr( "*Error(%d): Unknown Error.\r\n", iError );
 return;
}



int
DELIReadRegistrySettings(void)
{
 HKEY    hKey;
 DWORD   size;
 DWORD   type;
 char    szTmp[FILENAMESIZE];
 int     status;
 int     iTmp;

 status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                                        0, KEY_READ, &hKey);
 if ( status != ERROR_SUCCESS )
  return ERR_CANT_FIND_TPCC_KEY;

 size = sizeof(szTmp);
 status = RegQueryValueEx(hKey, "LoginDelay", 0, &type, szTmp, &size);
 if ( status == ERROR_SUCCESS )
  giLoginDelay = atoi(szTmp);

 size = sizeof(szTmp);
 status = RegQueryValueEx(hKey, "NumberOfDeliveryThreads",
                                          0, &type, szTmp, &size);
 if ( status == ERROR_SUCCESS )
  if ( 0 != ( iTmp = atoi(szTmp) ))
   iNumThreads = iTmp;

 size = sizeof(szTmp);
 status = RegQueryValueEx(hKey, "NumberOfQueuedDeliveryTransactions",
                                                0, &type, szTmp, &size);
 if ( status == ERROR_SUCCESS )
  if ( 0 != ( iTmp = atoi(szTmp) ))
   iNumQueued = iTmp;

 size = sizeof(szTmp);
 status = RegQueryValueEx(hKey, "DeadlockRetry", 0, &type, szTmp, &size);
 if ( status == ERROR_SUCCESS )
  if ( 0 != ( iTmp = atoi(szTmp) ))
   iDeadlockRetry = iTmp;

 size = sizeof(szTmp);
 status =RegQueryValueEx(hKey, "DeliveryServerConnectsToDB",
                                              0, &type, szTmp, &size);
 if ( status == ERROR_SUCCESS )
  if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
   bDirectConnect = TRUE;

 size = sizeof(szTmp);
 status =RegQueryValueEx(hKey, "FlushDeliveryLog", 0, &type, szTmp,
&size);
 if ( status == ERROR_SUCCESS )
  if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
   bFlush = TRUE;

 size = sizeof(szTmp);
 status =RegQueryValueEx(hKey, "DisplayDeliveryCompletions", 0, &type,
szTmp, &size);
 if ( status == ERROR_SUCCESS )
  if ( 0 == stricmp(szTmp, "FALSE") || 0 == strcmp(szTmp, "0"))
   gbDisplayCompletions = FALSE;

 size = sizeof(szTmp);
 status =RegQueryValueEx(hKey, "UseDeliveryPipe", 0, &type, szTmp, &size);
 if ( status == ERROR_SUCCESS )
  if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
   gbUsePipe = TRUE;

 size = sizeof(szTmp);
 status =RegQueryValueEx(hKey, "WriteDeliveryLog", 0, &type, szTmp,
&size);
 if ( status == ERROR_SUCCESS )
  if ( 0 == stricmp(szTmp, "TRUE") || 0 == strcmp(szTmp, "1"))
   bLog = TRUE;

 size = sizeof(szTmp);
 status = RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size);
 if ( status != ERROR_SUCCESS )
  return ERR_CANT_FIND_PATH_VALUE;
 if( bLog ) {
  strcpy(szLogPath, szTmp);
  strcat(szLogPath, "delilog.");
  fpLog = fopen(szLogPath, "wb");
  if ( NULL == fpLog )
   return ERR_CANNOT_CREATE_RESULTS_FILE;
 }

 RegCloseKey(hKey);

 return ERR_SUCCESS;
}



void
TPCCDeliveryDeferredResponse( int retcode, pDeliveryData pDelivery )
{
 size_t                                  bw;
 int                                             status;
```

```
 if ( ERR_DB_PENDING == retcode )
 {
  return;
 }
 else
 {



  if( ERR_DB_SUCCESS != retcode)
  {

   pDelivery->queue_time = 1;
   DELIErrorMessage(retcode);
  }
  pDelivery->delta_time = GetTickCount( ) - pDelivery->delta_time;


  if( bLog )
   DELILog( pDelivery );

  if( gbDisplayCompletions && !gbUsePipe ) {

   status = bufwrite( &pDelivery, sizeof(pDeliveryData),
                                        &bw, INFINITE, outputbuf );
   if( BUF_SUCCESS != status ) {
            DELIErrorMessage( ERR_DELIVERY_OUTPUT_PIPE_WRITE );
            return;
   }
  }
  else {

   UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS,
pDelivery );
  }
 }
}



unsigned __stdcall
DELIThread( void *ptr )
{
 size_t                  br;
 pDeliveryData                   pDelivery;
 int                             retcode;
 int                             CC;
 CallersContext          *pCC;
 LoginData               login;
 char                            tmp[16];
 int                             status;
 DBContext               DBC;


 CC = 0;
 pCC = &CC;
 DBC = INVALID_DB_CONTEXT;



 if( bDirectConnect ) {
  gethostname(tmp, sizeof(tmp));
  login.w_id = 0;
  login.ld_id = 0;
  login.pCC = pCC;
  strcpy( login.szServer, gszServer );
  strcpy( login.szDatabase, gszDatabase );
  strcpy( login.szUser, gszUser );
  strcpy( login.szPassword, gszPassword );
  sprintf( login.szApplication, "%s:delisrv - %d",
                tmp, GetCurrentThreadId());
  status = TPCCConnectDB( &DBC, &login );
  if( ERR_DB_SUCCESS != status && ERR_SUCCESS != giDeliInitStatus )
   giDeliInitStatus = status;
 }


 if ( InterlockedDecrement( &gDeliThreadStartCtr ) == 0 )
  SetEvent( gDeliThreadStartEvent );

 WaitForSingleObject( gDeliThreadStartEvent, INFINITE );

 if( ERR_SUCCESS != giDeliInitStatus )
  return giDeliInitStatus;


 while( !bDone ){

  if( gbUsePipe ) {
   RESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS, pDelivery );
   if( 0 == ReadFile( hPipeInputRead, pDelivery, sizeof( DeliveryDataInput ),
                                        &br, NULL )) {
            status = GetLastError( );
            DELIErrorMessage( ERR_DELIVERY_PIPE_READ );
            continue;
   }
  }
  else {
   if(BUF_SUCCESS != bufread( &pDelivery, sizeof( pDeliveryData ), &br,
                                                INFINITE, inputbuf )) {
            DELIErrorMessage( ERR_DELIVERY_PIPE_READ );
            continue;
   }

  }
  if( bDirectConnect )
   retcode = TPCCDeliveryDB( DBC, pDelivery );
  else
   retcode = TPCCDeliveryDeferred( pDelivery );

  TPCCDeliveryDeferredResponse( retcode, pDelivery );
 }

 return ERR_SUCCESS;
}



void
DELILog( pDeliveryData pDelivery )
{
 struct tm               start;
 struct tm               *end;
 time_t                  endt;
 unsigned                delta_time_seconds;
 unsigned                delta_time_milliseconds;

 delta_time_seconds = pDelivery->delta_time / 1000;
 delta_time_milliseconds = pDelivery->delta_time - (delta_time_seconds *
1000);

 CopyMemory( &start, localtime( &pDelivery->queue_time ), sizeof( start ));
 endt = pDelivery->queue_time + delta_time_seconds;
 end = localtime( &endt );

 fprintf( fpLog,
            "%2.2d/%2.2d/%2.2d,"
            "%2.2d:%2.2d:%2.2d:%3.3d,"
            "%2.2d:%2.2d:%2.2d:%3.3d,"
            "%8.8d,"
            "%5.5d,%2.2d,"
            "%4.4d,%4.4d,%4.4d,%4.4d,%4.4d,"
            "%4.4d,%4.4d,%4.4d,%4.4d,%4.4d\r\n",
            start.tm_year, start.tm_mon, start.tm_mday,
```

```
                start.tm_hour, start.tm_min, start.tm_sec, 0,
                end->tm_hour, end->tm_min, end->tm_sec,
delta_time_milliseconds,
                pDelivery->delta_time,
                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);

 return;
}
```

deli_srv.h

```
#ifndef DELI_SRV_H
#define DELI_SRV_H


/*+*********************************************************************
 *                                                                     *
 * COPYRIGHT (c) 1997 BY                                               *
 * COMPAQ COMPUTER CORPORATION, .                                      *
 * ALL RIGHTS RESERVED.                                                *
 *                                                                     *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  *
 * TRANSFERRED.                                                        *
 *                                                                     *
 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  *
 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY COMPAQ COMPUTER  *
 * CORPORATION.                                                        *
 *                                                                     *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.              *
 *                                                                     *
 *                                                                     *
 **********************************************************************/

/*+
 * Abstract: This file contains definitions of the delivery server queue.
 *
 * Author:  W Carr
 * Creation Date: July 1997
 *
 * Modified history:
 *              21-Oct-1997 WCarr       Replaced NamedPipe with buffer code.
 *
*/

#ifdef DELI_SRV_C
# define DELI_GLOBAL(thing,init) thing = init
#else
# define DELI_GLOBAL(thing,init) extern thing
#endif

#if defined DELI_CLI_C || defined DELI_SRV_C
DELI_GLOBAL(BOOL gbDisplayCompletions,TRUE);
DELI_GLOBAL(BUFPTR inputbuf,NULL);
DELI_GLOBAL(BUFPTR outputbuf,NULL);


DELI_GLOBAL(BOOL gbUsePipe,FALSE);
DELI_GLOBAL(HANDLE ghPipeInputWrite,INVALID_HANDLE_VALUE);
#endif

int DELIGetTransportData( BOOL *dy_use_transport,
                          int *num_dy_servers,
                          int *num_queued_deliveries,
                          int *num_queued_responses );
int DELIServerStartup( BOOL dy_use_transport, int num_dy_servers,
                       int num_queued_deliveries, int
num_queued_responses );
int DELIServerShutdown( void );

#endif
```

tpcc.c

```
/*+          FILE:       TPCC.C
 *                       Microsoft TPC-C Kit Ver. 3.00.000
 *                       Audited 08/23/96            By Francois Raab
 *
 *                       Copyright Microsoft, 1996
 *                       Copyright Compaq Computer Corp., 1997
 *
 *          PURPOSE:    Main module for TPCC.DLL which is an ISAPI service dll.
 *          Author:             Philip Durr
 *                                          philipdu@Microsoft.com
 *
 *          MODIFICATIONS:
 *
 *                       Routines substantially modified by:
 *                                  Anne Bradley  Compaq Computer Corp.
 *                                  Bill Carr     Compaq Computer Corp.
 *
 */
/*+*********************************************************************
 *                                                                     *
 * COPYRIGHT (c) 1997 BY                                               *
 * COMPAQ COMPUTER CORPORATION, .                                      *
 * ALL RIGHTS RESERVED.                                                *
 *                                                                     *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE  *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER  *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY  *
 * TRANSFERRED.                                                        *
 *                                                                     *
 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE  *
 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY COMPAQ COMPUTER  *
 * CORPORATION.                                                        *
 *                                                                     *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.              *
 *                                                                     *
 *                                                                     *
 **********************************************************************/


#include <windows.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>

#define TPCC_C

#include <tpccerr.h>
#include <tpccstruct.h>
#include <tpccapi.h>
#include <httpext.h>

#include <tpcc.h>
#include <web_ui.h>
```

```
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;
}



int ParseNewOrderQuery(char *pQueryString, NewOrderData
*pNewOrderData)
{
 char   *ptr;
 int                    i;
 short      items;
 BOOL       bCheck;
 char       *pProcessedQuery[MAXNEWORDERVALS];


 PARSE_QUERY_STRING(pQueryString, MAXNEWORDERVALS,
                        newOrderStrs, pProcessedQuery);

 if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
  return ERR_NEWORDER_FORM_MISSING_DID;

 GetNumeric(ptr, &pNewOrderData->d_id);
 if(0 == pNewOrderData->d_id)
  return ERR_NEWORDER_DISTRICT_INVALID;


 if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
  return ERR_NEWORDER_CUSTOMER_KEY;

 if( !GetNumeric(ptr, &pNewOrderData->c_id))
  return ERR_NEWORDER_CUSTOMER_INVALID;

 bCheck = FALSE;
 pNewOrderData->o_all_local = 1;


 for(i=0, items=0; i<15; i++)
 {
  if( !GetValuePtr(pProcessedQuery, i*3+IID00, &ptr))
   return ERR_NEWORDER_MISSING_IID_KEY;
  if(*ptr != '&' && *ptr)
  {

   if ( bCheck )
        return ERR_NEWORDER_ITEM_BLANK_LINES;
   if(!GetNumeric(ptr, &pNewOrderData->o_ol[i].ol_i_id))
        return ERR_NEWORDER_ITEMID_INVALID;

   if(!GetValuePtr(pProcessedQuery, i*3+SP00, &ptr))
        return ERR_NEWORDER_MISSING_SUPPW_KEY;
   if(!GetNumeric(ptr, &pNewOrderData->o_ol[i].ol_supply_w_id))
        return ERR_NEWORDER_SUPPW_INVALID;
   if ( pNewOrderData->o_all_local &&
        pNewOrderData->o_ol[i].ol_supply_w_id !=
        pNewOrderData->w_id )
        pNewOrderData->o_all_local = 0;
   if(!GetValuePtr(pProcessedQuery, i*3+QTY00, &ptr))
        return ERR_NEWORDER_MISSING_QTY_KEY;
   if(!GetNumeric(ptr, &pNewOrderData->o_ol[i].ol_quantity))
        return ERR_NEWORDER_QTY_INVALID;
   items++;
   if ( pNewOrderData->o_ol[i].ol_i_id >= 1000000 ||
        pNewOrderData->o_ol[i].ol_i_id < 1 )
        return ERR_NEWORDER_ITEMID_RANGE;
   if ( pNewOrderData->o_ol[i].ol_quantity >= 100 ||
        pNewOrderData->o_ol[i].ol_quantity < 1 )
        return ERR_NEWORDER_QTY_RANGE;
  }
  else
  {
   if(!GetValuePtr(pProcessedQuery, i*3+SP00, &ptr))
        return ERR_NEWORDER_MISSING_QTY_KEY;
   if(*ptr != '&' && *ptr)
        return ERR_NEWORDER_SUPPW_WITHOUT_ITEMID;

   if(!GetValuePtr(pProcessedQuery, i*3+QTY00, &ptr))
        return ERR_NEWORDER_MISSING_QTY_KEY;
   if(*ptr != '&' && *ptr)
        return ERR_NEWORDER_QTY_WITHOUT_ITEMID;
   bCheck = TRUE;
  }
 }
 if ( items == 0 )
  return ERR_NEWORDER_NOITEMS_ENTERED;

 pNewOrderData->o_ol_cnt = items;

 return ERR_SUCCESS;
}


int ParseOrderStatusQuery(char *pQueryString,
                                OrderStatusData *pOrderStatusData)
{
 char       szTmp[26];
 char   *ptr;
 char       *pSzTmp;
 char       *pProcessedQuery[MAXORDERSTATUSVALS];


 PARSE_QUERY_STRING(pQueryString, MAXORDERSTATUSVALS,
                        orderStatusStrs, pProcessedQuery);

 if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
  return ERR_ORDERSTATUS_MISSING_DID_KEY;
 if ( !GetNumeric(ptr, &pOrderStatusData->d_id) )
  return ERR_ORDERSTATUS_DID_INVALID;

 if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
  return ERR_ORDERSTATUS_MISSING_CID_KEY;

 if ( *ptr == '&' || !(*ptr))
 {
  pSzTmp = szTmp;
  pOrderStatusData->c_id = 0;
  if ( !GetValuePtr(pProcessedQuery, CLT_O, &ptr) )
   return ERR_ORDERSTATUS_MISSING_CLT_KEY;
  while(*ptr != '&' && *ptr)
  {
   *pSzTmp = *ptr;
   pSzTmp++;
   ptr++;
  }
  *pSzTmp = '\0';
  _strupr( szTmp );
  strcpy(pOrderStatusData->c_last, szTmp);
  if ( strlen(pOrderStatusData->c_last) > 16 )
```

```
    return ERR_ORDERSTATUS_CLT_RANGE;
  }
  else
  {
   if (!GetNumeric(ptr, &pOrderStatusData->c_id))
    return ERR_ORDERSTATUS_CID_INVALID;
   if ( !GetValuePtr(pProcessedQuery, CLT_O, &ptr) )
    return ERR_ORDERSTATUS_MISSING_CLT_KEY;
   if ( *ptr != '&' && *ptr)
    return ERR_ORDERSTATUS_CID_AND_CLT;
  }

  return ERR_SUCCESS;
}



int ParsePaymentQuery(char *pQueryString, PaymentData *pPaymentData)
{
  char          szTmp[26];
  char          *ptr;
  char    *pPtr;
  char          *pSzTmp;
  char          *pProcessedQuery[MAXPAYMENTVALS];


  PARSE_QUERY_STRING(pQueryString, MAXPAYMENTVALS,
                                  paymentStrs, pProcessedQuery);

  if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
   return ERR_PAYMENT_MISSING_DID_KEY;
  if ( !GetNumeric(ptr, &pPaymentData->d_id) )
   return ERR_PAYMENT_DISTRICT_INVALID;

  if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
   return ERR_PAYMENT_MISSING_CID_KEY;

  pPaymentData->c_id = 0;
  if ( (*ptr != '&' && *ptr) && !GetNumeric(ptr, &pPaymentData->c_id) )
   return ERR_PAYMENT_CUSTOMER_INVALID;


  if(*ptr == '&' || !(*ptr))
  {
   pSzTmp = szTmp;
   if ( !GetValuePtr(pProcessedQuery, CLT_P, &ptr) )
    return ERR_PAYMENT_MISSING_CLT;
   while(*ptr != '&' && *ptr)
   {
    *pSzTmp = *ptr;
    pSzTmp++;
    ptr++;
   }
   *pSzTmp = '\0';
   _strupr( szTmp );

   strcpy(pPaymentData->c_last, szTmp);
   if ( strlen(pPaymentData->c_last) > 16 )
    return ERR_PAYMENT_LAST_NAME_TO_LONG;
  }
  else
  {
   if ( !GetValuePtr(pProcessedQuery, CLT_P, &ptr) )
    return ERR_PAYMENT_MISSING_CLT_KEY;
   if(*ptr != '&' && *ptr)
    return ERR_PAYMENT_CID_AND_CLT;
  }

  if ( !GetValuePtr(pProcessedQuery, CDI, &ptr) )
   return ERR_PAYMENT_MISSING_CDI_KEY;
  if ( !GetNumeric(ptr, &pPaymentData->c_d_id) )
   return ERR_PAYMENT_CDI_INVALID;

  if ( !GetValuePtr(pProcessedQuery, CWI, &ptr) )
   return ERR_PAYMENT_MISSING_CWI_KEY;

  if ( !GetNumeric(ptr, &pPaymentData->c_w_id) )
   return ERR_PAYMENT_CWI_INVALID;



  if ( !GetValuePtr(pProcessedQuery, HAM, &ptr) )
   return ERR_PAYMENT_MISSING_HAM_KEY;

  pPtr = ptr;
  while( *pPtr != '&' && *pPtr)
  {
   if ( *pPtr == '.' )
   {
    pPtr++;
    if ( !*pPtr )
          break;
    if ( *pPtr < '0' || *pPtr > '9' )
          return ERR_PAYMENT_HAM_INVALID;
    pPtr++;
    if ( !*pPtr )
          break;
    if ( *pPtr < '0' || *pPtr > '9' )
          return ERR_PAYMENT_HAM_INVALID;
    if ( !*pPtr )
          return ERR_PAYMENT_HAM_INVALID;
   }
   else if ( *pPtr < '0' || *pPtr > '9' )
    return ERR_PAYMENT_HAM_INVALID;
   pPtr++;
  }

  pPaymentData->h_amount = atof(ptr);
  if ( pPaymentData->h_amount >= 10000.00 || pPaymentData->h_amount < 0 )
   return ERR_PAYMENT_HAM_RANGE;

  return ERR_SUCCESS;
}



int ReadRegistrySettings(void)
{
  HKEY      hKey;
  DWORD     size;
  DWORD     type;
  char          szTmp[FILENAMESIZE];
  int   status;
  int           iTmp;


  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                                        0, KEY_READ, &hKey);
  if( status != ERROR_SUCCESS )
   return ERR_CANT_FIND_TPCC_KEY;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
   return ERR_CANT_FIND_PATH_VALUE;
  strcpy(szTpccLogPath, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Server", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

   return ERR_CANT_FIND_SERVER_VALUE;
  strcpy(gszServer, szTmp);
```

```
  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Database", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_DATABASE_VALUE;
  strcpy(gszDatabase, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "User", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_USER_VALUE;
  strcpy(gszUser, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Password", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_PASSWORD_VALUE;
  strcpy(gszPassword, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "CkptUser", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    strcpy(gszCkptUser, gszUser);
  else
    strcpy(gszCkptUser, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "CkptPassword", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    strcpy(gszCkptPassword, gszPassword);
  else
    strcpy(gszCkptPassword, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "LOG", 0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS  && 0 == stricmp(szTmp, "ON") )
    bLog = TRUE;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "MaximumWarehouses", 0, &type, szTmp,
&size);
  if ( status == ERROR_SUCCESS  && 0 != (iTmp = atoi(szTmp)) )
    iMaxWareHouses = iTmp;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "MaxConnections", 0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS && 0 != (iTmp = atoi(szTmp)) )
    iMaxConnections = iTmp;

  RegCloseKey(hKey);

  if( RegOpenKeyEx(HKEY_LOCAL_MACHINE,

"SYSTEM\\CurrentControlSet\\Services\\InetInfo\\Parameters",
                              0, KEY_READ, &hKey) != ERROR_SUCCESS )
    return ERR_CANT_FIND_INETINFO_KEY;

  size = sizeof(gdwPoolThreadLimit);
  if ( RegQueryValueEx(hKey, "PoolThreadLimit", 0, &type,
                                (LPBYTE)&gdwPoolThreadLimit, &size) !=
ERROR_SUCCESS )
    return ERR_CANT_FIND_POOLTHREADLIMIT;

  RegCloseKey(hKey);

  return ERR_SUCCESS;

}
```

tpcc.h

```
#ifndef TPCC_H
#define TPCC_H

/*+**************************************************************************
 *                                                                          *
 * COPYRIGHT (c) 1997 BY                                                    *
 * COMPAQ COMPUTER CORPORATION, .                                           *
 * ALL RIGHTS RESERVED.                                                     *
 *                                                                          *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
 * ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
 * OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
 * TRANSFERRED.                                                             *
 *                                                                          *
 * THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
 * AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY COMPAQ COMPUTER      *
 * CORPORATION.                                                             *
 *                                                                          *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                   *
 *                                                                          *
 *                                                                          *
 **************************************************************************/

/*+
 * Abstract: This is the header file for web_ui.c.  it contains the
 *           function prototypes for the routines that are called outside web_ui.c
 *
 * Author: A Bradley
 * Creation Date: May 1997
 *
 * Modified history:
 *
 *
 */


#define FILENAMESIZE 256

#if defined WEB_UI_C || defined TPCC_C

void FormatString(char *szDest, char *szPic, char *szSrc);

int ParseNewOrderQuery(char *pQueryString, NewOrderData
*pNewOrderData);
int ParsePaymentQuery(char *pQueryString, PaymentData *pPaymentData);
int ParseOrderStatusQuery(char *pQueryString,
                                        OrderStatusData *pOrderStatusData);
#endif

BOOL ReadRegistrySettings(void);


#ifdef TPCC_C
#define GLOBAL(thing,initializer) thing = initializer
#else
#define GLOBAL(thing,initializer) extern thing
#endif

#if defined WEB_UI_C || defined TPCC_C
#endif
GLOBAL(int iMaxConnections,25);
GLOBAL(BOOL bLog,FALSE);
GLOBAL(int iDeadlockRetry,3);
GLOBAL(char szTpccLogPath[FILENAMESIZE],{'\0'});
GLOBAL(int iMaxWareHouses,500);
GLOBAL(char gszServer[32],{'\0'});
GLOBAL(char gszDatabase[32],"tpcc");
GLOBAL(char gszUser[32],"sa");
```

```
GLOBAL(char gszPassword[32],{'\0'});
GLOBAL(char gszCkptUser[32],"sa");
GLOBAL(char gszCkptPassword[32],{'\0'});
GLOBAL(pTransactionPoolStruct gpTransactionPool,{0});

#endif
```

tpcc_acmsxp.h

```
/*+*************************************************************************
 *                                                                         *
 * COPYRIGHT (c) 1997 BY                                                   *
 * COMPAQ COMPUTER CORPORATION, .                                          *
 * ALL RIGHTS RESERVED.                                                    *
 *                                                                         *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE          *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER     *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
 * TRANSFERRED.                                                            *
 *                                                                         *
 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE    *
 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY COMPAQ COMPUTER          *
 * CORPORATION.                                                            *
 *                                                                         *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                  *
 *                                                                         *
 *                                                                         *
 **************************************************************************/

#ifndef TPCC_ACMSXP_H
#define TPCC_ACMSXP_H

#include <tpccstruct.h>


#define TYPE_NO 1
#define TYPE_OS 2
#define TYPE_PT 3
#define TYPE_SL 4
#define TYPE_GC 5
#define TYPE_DY 6

int ACMSxpStartup( int maxDBConnections, long totalGovValue );
int ACMSxpShutdown( void );
BOOL getGovernorValue( char *group, long *ctr );


#endif
```

tpcc_acmsxp_pp.stdl

```
#Cinclude "tpcc_stdl_workspaces"
!
! STDL DY Processing group definition for TPC-C
!
PROCESSING GROUP tpcc_dy
  UUID "11111111-8322-11cf-1111-08002be44c83";
  VERSION 1.0;
  SOURCE LANGUAGE IS C;

PROCEDURE dy_transaction_init
  ARGUMENT IS
                io_login_wksp PASSED AS INOUT;

PROCEDURE dy_transaction
  ARGUMENT IS
                io_dy_wksp PASSED AS INOUT;

END PROCESSING GROUP;

!
! STDL NO Processing group definition for TPC-C
!
PROCESSING GROUP tpcc_no
  UUID "2402d1dc-8322-11cf-ba68-08002be44c83";
  VERSION 1.0;
  SOURCE LANGUAGE IS C;

PROCEDURE no_transaction_init
  ARGUMENT IS
                io_login_wksp PASSED AS INOUT;

PROCEDURE no_transaction
  ARGUMENT IS
                io_no_wksp PASSED AS INOUT;

END PROCESSING GROUP;

!
! STDL OS Processing group definition for TPC-C
!
PROCESSING GROUP tpcc_os
  UUID "4d654a5f-8322-11cf-b420-08002be44c83";
  VERSION 1.0;
  SOURCE LANGUAGE IS C;

PROCEDURE os_transaction_init
  ARGUMENT IS
                io_login_wksp PASSED AS INOUT;

PROCEDURE os_transaction
  ARGUMENT IS
                io_os_wksp PASSED AS INOUT;

END PROCESSING GROUP;

!
! STDL PT Processing group definition for TPC-C
!
PROCESSING GROUP tpcc_pt
  UUID "70e38ff0-8322-11cf-99f7-08002be44c83";
  VERSION 1.0;
  SOURCE LANGUAGE IS C;

PROCEDURE pt_transaction_init
  ARGUMENT IS
                io_login_wksp PASSED AS INOUT;

PROCEDURE pt_transaction
  ARGUMENT IS
                io_pt_wksp PASSED AS INOUT;

END PROCESSING GROUP;

!
! STDL SL Processing group definition for TPC-C
!
PROCESSING GROUP tpcc_sl
  UUID "c02bdfc2-8322-11cf-b76e-08002be44c83";
  VERSION 1.0;
  SOURCE LANGUAGE IS C;

PROCEDURE sl_transaction_init
  ARGUMENT IS
                io_login_wksp PASSED AS INOUT;

PROCEDURE sl_transaction
  ARGUMENT IS
                io_sl_wksp PASSED AS INOUT;
```

```
END PROCESSING GROUP;

!
! STDL GC Processing group definition for TPC-C
!
PROCESSING GROUP tpcc_gc
  UUID "00000000-8322-11cf-0000-08002be44c83";
  VERSION 1.0;
  SOURCE LANGUAGE IS C;

PROCEDURE gc_transaction_init
  ARGUMENT IS
                io_login_wksp PASSED AS INOUT;

PROCEDURE gc_transaction
  ARGUMENT IS
                io_gc_wksp PASSED AS INOUT,
                int_gc_wksp PASSED AS INPUT;

END PROCESSING GROUP;
```

tpcc_fct.c

```
/*+*********************************************************************
 *                                                                     *
 *  COPYRIGHT (c) 1997 BY                                              *
 *  COMPAQ COMPUTER CORPORATION, .                                     *
 *  ALL RIGHTS RESERVED.                                               *
 *                                                                     *
 *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
 *  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
 *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
 *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
 *  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
 *  TRANSFERRED.                                                       *
 *                                                                     *
 *  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
 *  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY COMPAQ COMPUTER  *
 *  CORPORATION.                                                       *
 *                                                                     *
 *  COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.             *
 *                                                                     *
 *                                                                     *
 ***********************************************************************/

#include <windows.h>
#include <string.h>
#include <stdio.h>
#include <process.h>
#include <stdldefs.h>
#define STDL_SYNCHRONOUS ((void*)0xffffffff)

#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

#include <tpcc_acmsxp.h>
#include <tpcc_acmsxp_pp.h>

#include <tpcc.h>

#include <deli_cli.h>
#include <deli_srv.h>

#ifdef FFE_DEBUG
# include <crtdbg.h>
#endif

#define TOTAL_ADMIN_CONNECTIONS 1

#define FILENAMESIZE 256

static HANDLE gForceAllThreadsEvent;
static int    gForceAllThreadsCtr;
static int    gInitRetStatus;
static int    giLoginDelay = 0;
static BOOL   gbGeneric = FALSE;

void _STDL_CALL_ dy_callback( void *);
void _STDL_CALL_ no_callback( void *);
void _STDL_CALL_ os_callback( void *);
void _STDL_CALL_ pt_callback( void *);
void _STDL_CALL_ sl_callback( void *);

struct int_gc_wksp dy = { TYPE_DY };
struct int_gc_wksp no = { TYPE_NO };
struct int_gc_wksp os = { TYPE_OS };
struct int_gc_wksp pt = { TYPE_PT };
struct int_gc_wksp sl = { TYPE_SL };
struct int_gc_wksp gc = { TYPE_GC };

typedef struct {
  int type;
} force_connect_args;

void __cdecl force_connect( void *arglist );
BOOL getGovernorValue( char *group, long *ctr );
#ifdef USE_MUX
stdl_call_dll_init( DWORD why_called );
#endif

#ifdef FFE_DEBUG
pDeliveryData gpDelivery = NULL;
pNewOrderData gpNewOrder = NULL;
pOrderStatusData gpOrderStatus = NULL;
pPaymentData gpPayment = NULL;
pStockLevelData gpStockLevel = NULL;
#endif

void _STDL_CALL_
dy_callback( void *data )
{
  pDeliveryData pDelivery = data;

#ifdef FFE_DEBUG
   _ASSERT( VALID_DB_ERR(pDelivery->status) );
#endif

  TPCCDeliveryDeferredResponse( pDelivery->status, pDelivery );



}
void _STDL_CALL_
no_callback( void *data )
{
  pNewOrderData pNewOrder = data;
  pCallersContext pCC;

#ifdef FFE_DEBUG
   _ASSERT( VALID_DB_ERR(pNewOrder->status) );
#endif

  pCC = pNewOrder->pCC;

  TPCCNewOrderResponse( pNewOrder->status, pNewOrder );

  TPCCResponseComplete( pCC );
}
void _STDL_CALL_
os_callback( void *data )
{
  pOrderStatusData pOrderStatus = data;
  pCallersContext pCC;
```

```
#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(pOrderStatus->status) );
#endif

  pCC = pOrderStatus->pCC;

  TPCCOrderStatusResponse( pOrderStatus->status, pOrderStatus );

  TPCCResponseComplete( pCC );
}
void _STDL_CALL_
pt_callback( void *data )
{
  pPaymentData pPayment = data;
  pCallersContext pCC;

#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(pPayment->status) );
#endif

  pCC = pPayment->pCC;

  TPCCPaymentResponse( pPayment->status, pPayment );

  TPCCResponseComplete( pCC );
}
void _STDL_CALL_
sl_callback( void *data )
{
  pStockLevelData pStockLevel = data;
  pCallersContext pCC;

#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(pStockLevel->status) );
#endif

  pCC = pStockLevel->pCC;

  TPCCStockLevelResponse( pStockLevel->status, pStockLevel );

  TPCCResponseComplete( pCC );
}


int
TPCCDeliveryDeferred( pDeliveryData ptr )
{
  int retcode = ERR_DB_PENDING;

#ifdef FFE_DEBUG
  ptr->iStage |= IN_LH;
  ptr->iSynchronous = 0;
#endif


  if( 2 == gbGeneric ) {
#ifdef FFE_DEBUG
  ptr->iSynchronous = 1;
#endif
    gc_transaction( STDL_SYNCHRONOUS, NULL, (struct io_gc_wksp *) ptr,
&dy );
    retcode = ptr->status;
  }
  else if( gbGeneric )
    gc_transaction( dy_callback, (void *)ptr, (struct io_gc_wksp *) ptr, &dy );
  else
    dy_transaction( dy_callback, (void *)ptr, (struct io_dy_wksp *) ptr );

#ifdef FFE_DEBUG
  ptr->iStage |= LEAVING_LH;
  _ASSERT( VALID_DB_ERR(retcode));
#endif

  return retcode;
}


int
TPCCNewOrder( pNewOrderData ptr )
{
  int retcode = ERR_DB_PENDING;

#ifdef FFE_DEBUG
  ptr->iStage |= IN_LH;
  ptr->iSynchronous = 0;
#endif


  if( 2 == gbGeneric ) {
#ifdef FFE_DEBUG
  ptr->iSynchronous = 1;
#endif
    gc_transaction( STDL_SYNCHRONOUS, NULL, (struct io_gc_wksp *) ptr,
&no );
    retcode = ptr->status;
  }
  else if( gbGeneric )
    gc_transaction( no_callback, (void *)ptr, (struct io_gc_wksp *) ptr, &no );
  else
    no_transaction( no_callback, (void *)ptr, (struct io_no_wksp *) ptr );

#ifdef FFE_DEBUG
  ptr->iStage |= LEAVING_LH;
  _ASSERT( VALID_DB_ERR(retcode));
#endif

  return retcode;
}


int
TPCCOrderStatus( pOrderStatusData ptr )
{
  int retcode = ERR_DB_PENDING;

#ifdef FFE_DEBUG
  ptr->iStage |= IN_LH;
  ptr->iSynchronous = 0;
#endif


  if( 2 == gbGeneric ) {
#ifdef FFE_DEBUG
  ptr->iSynchronous = 1;
#endif
    gc_transaction( STDL_SYNCHRONOUS, NULL, (struct io_gc_wksp *) ptr,
&os );
    retcode = ptr->status;
  }
  else if( gbGeneric )
    gc_transaction( os_callback, (void *)ptr, (struct io_gc_wksp *) ptr, &os );
  else
    os_transaction( os_callback, (void *)ptr, (struct io_os_wksp *) ptr );

#ifdef FFE_DEBUG
  ptr->iStage |= LEAVING_LH;
  _ASSERT( VALID_DB_ERR(retcode));
#endif

  return retcode;
}


int
```

```
TPCCPayment( pPaymentData ptr )
{
 int retcode = ERR_DB_PENDING;

#ifdef FFE_DEBUG
 ptr->iStage |= IN_LH;
 ptr->iSynchronous = 0;
#endif


 if( 2 == gbGeneric ) {
#ifdef FFE_DEBUG
 ptr->iSynchronous = 1;
#endif
   gc_transaction( STDL_SYNCHRONOUS, NULL, (struct io_gc_wksp *) ptr,
&pt );
   retcode = ptr->status;
 }
 else if( gbGeneric )
   gc_transaction( pt_callback, (void *)ptr, (struct io_gc_wksp *) ptr, &pt );
 else
   pt_transaction( pt_callback, (void *)ptr, (struct io_pt_wksp *) ptr );

#ifdef FFE_DEBUG
 ptr->iStage |= LEAVING_LH;
 _ASSERT( VALID_DB_ERR(retcode));
#endif

 return retcode;
}


int
TPCCStockLevel( pStockLevelData ptr )
{
 int retcode = ERR_DB_PENDING;

#ifdef FFE_DEBUG
 ptr->iStage |= IN_LH;
 ptr->iSynchronous = 0;
#endif


 if( 2 == gbGeneric ) {
#ifdef FFE_DEBUG
 ptr->iSynchronous = 1;
#endif
   gc_transaction( STDL_SYNCHRONOUS, NULL, (struct io_gc_wksp *) ptr,
&sl );
   retcode = ptr->status;
 }
 else if( gbGeneric )
   gc_transaction( sl_callback, (void *)ptr, (struct io_gc_wksp *) ptr, &sl );
 else
   sl_transaction( sl_callback, (void *)ptr, (struct io_sl_wksp *) ptr );

#ifdef FFE_DEBUG
 ptr->iStage |= LEAVING_LH;
 _ASSERT( VALID_DB_ERR(retcode));
#endif

 return retcode;
}


BOOL
getGovernorValue( char *group, long *ctr )
{
 HKEY      hKey;
 DWORD   size;
 DWORD   type;
 char        keyname[256];
 BOOL      status;

 sprintf( keyname,
          "SOFTWARE\\Compaq Computer Corporation\\TPware\\Group
Settings\\%s",
           group );

 if( ERROR_SUCCESS != RegOpenKeyEx( HKEY_LOCAL_MACHINE,
keyname,
                                    0, KEY_READ, &hKey ))
{
  TPCCErr( "Error, group settings not found for %s\r\n", keyname );
  return FALSE;
 }

 size = sizeof( *ctr );
 if ( ERROR_SUCCESS ==
    RegQueryValueEx( hKey, "MaxThreads", 0, &type, (LPBYTE) ctr, &size ))
{
  status = TRUE;
 }
 else {
  TPCCErr( "Error, MaxThreads value not found for key %s\r\n", keyname );
  status = FALSE;
 }

 RegCloseKey( hKey );

 return status;
}


int
TPCCGetTransportData( pTransportData pTransport )
{
 HKEY      hKey;
 DWORD   size;
 DWORD   type;
 char        szTmp[FILENAMESIZE];
 int   status;
 long  total;
 long dyGovValue;
 long noGovValue;
 long osGovValue;
 long ptGovValue;
 long slGovValue;
 long gcGovValue;

 memset( pTransport, 0, sizeof( *pTransport ));
 pTransport->asynchronous = TRUE;

 status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                       0, KEY_READ, &hKey);
 if ( status != ERROR_SUCCESS )
  return ERR_CANT_FIND_TPCC_KEY;

 size = sizeof(szTmp);
 status = RegQueryValueEx(hKey, "GenericTransactionServers",
                          0, &type, szTmp, &size);
 if ( status == ERROR_SUCCESS )
  gbGeneric = atoi(szTmp);


 RegCloseKey(hKey);


 if( !gbGeneric ) {
  pTransport->generic = FALSE;
  total = 0;
  if ( ! getGovernorValue( "tpcc_dy", &dyGovValue ))
   return ERR_GOVERNOR_VALUE_NOT_FOUND;
```

```
   total += dyGovValue;
   pTransport->num_dy = dyGovValue;
   if ( ! getGovernorValue( "tpcc_no", &noGovValue ))
     return ERR_GOVERNOR_VALUE_NOT_FOUND;
   total += noGovValue;
   pTransport->num_no = noGovValue;
   if ( ! getGovernorValue( "tpcc_os", &osGovValue ))
     return ERR_GOVERNOR_VALUE_NOT_FOUND;
   total += osGovValue;
   pTransport->num_os = osGovValue;
   if ( ! getGovernorValue( "tpcc_pt", &ptGovValue ))
     return ERR_GOVERNOR_VALUE_NOT_FOUND;
   total += ptGovValue;
   pTransport->num_pt = ptGovValue;
   if ( ! getGovernorValue( "tpcc_sl", &slGovValue ))
     return ERR_GOVERNOR_VALUE_NOT_FOUND;
   total += slGovValue;
   pTransport->num_sl = slGovValue;
  }
  else {
   pTransport->generic = TRUE;
   if ( ! getGovernorValue( "tpcc_gc", &gcGovValue ))
     return ERR_GOVERNOR_VALUE_NOT_FOUND;
   total = gcGovValue;
   pTransport->num_gc = gcGovValue;
  }

  status = DELIGetTransportData( &pTransport->dy_use_transport,
                                                  &pTransport-
>num_dy_servers,
                                                  &pTransport-
>num_queued_deliveries,
                                                  &pTransport-
>num_queued_responses );

  if( ERR_SUCCESS != status )
    return status;

  return ERR_SUCCESS;
}


int
TPCCStartup( int iNumUsers, pTransportData pTransport )
{
  HKEY      hKey;
  DWORD     size;
  DWORD     type;
  char      szTmp[FILENAMESIZE];
  int   status;
  unsigned long ulhThread;
  force_connect_args fca;
  int counts[5];
  int count;
  long total;
  int maxDBConnections;

#ifdef USE_MUX
  stdl_call_dll_init( DLL_PROCESS_ATTACH );
#endif

  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                                        0, KEY_READ, &hKey);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_TPCC_KEY;


  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Server", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_SERVER_VALUE;
  strcpy(gszServer, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Database", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_DATABASE_VALUE;
  strcpy(gszDatabase, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "User", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_USER_VALUE;
  strcpy(gszUser, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Password", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_PASSWORD_VALUE;
  strcpy(gszPassword, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "LoginDelay", 0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS )
    giLoginDelay = atoi(szTmp);

  RegCloseKey(hKey);


  total = 0;
  if( !gbGeneric ) {
    counts[TYPE_DY] = pTransport->num_dy;
    total += pTransport->num_dy;
    counts[TYPE_NO] = pTransport->num_no;
    total += pTransport->num_no;
    counts[TYPE_OS] = pTransport->num_os;
    total += pTransport->num_os;
    counts[TYPE_PT] = pTransport->num_pt;
    total += pTransport->num_pt;
    counts[TYPE_SL] = pTransport->num_sl;
    total += pTransport->num_sl;
  }
  else
    total = pTransport->num_gc;

  maxDBConnections = total + TOTAL_ADMIN_CONNECTIONS;
  if( !pTransport->dy_use_transport )
    maxDBConnections += pTransport->num_dy_servers;

  status = ACMSxpStartup( maxDBConnections, total );
  if ( status != ERR_DB_SUCCESS )
    return status;


  status = DELIServerStartup( pTransport->dy_use_transport,
                                          pTransport->num_dy_servers,
                                          pTransport->num_queued_deliveries,
                                          pTransport->num_queued_responses );
  if( ERR_SUCCESS != status )
    return status;

  status = DELIClientStartup( );
  if( ERR_SUCCESS != status )
    return status;

  gInitRetStatus = ERR_DB_SUCCESS;
  gForceAllThreadsCtr = total;
  gForceAllThreadsEvent = CreateEvent( 0, TRUE, FALSE, 0 );
```

```
 if ( NULL == gForceAllThreadsEvent ) return
ERR_CANT_CREATE_ALL_THREADS_EVENT;

 if( !gbGeneric ) {
  for( fca.type = TYPE_NO; fca.type <= TYPE_SL; fca.type++ ) {
   for( count = 0; count < counts[fca.type]; count++ ) {
                ulhThread = _beginthread( force_connect, 0, (void *) fca.type );
                if( -1 == ulhThread ) {
                  return ERR_CANT_START_FRCDINIT_THREAD;
                }
                Sleep( giLoginDelay );
   }
  }
 }
 else {
  for( count = 0; count < pTransport->num_gc; count++ ) {
   ulhThread = _beginthread( force_connect, 0, (void *) TYPE_GC );
   if( -1 == ulhThread ) {
                return ERR_CANT_START_FRCDINIT_THREAD;
   }
   Sleep( giLoginDelay );
  }
 }

 WaitForSingleObject( gForceAllThreadsEvent, INFINITE );

 return gInitRetStatus;
}



int
TPCCConnect( pLoginData pLogin )
{

 if( 0 != strcmp( pLogin->szServer, gszServer ))
  return ERR_SERVER_MISMATCH;

 if( 0 != strcmp( pLogin->szDatabase, gszDatabase ))
  return ERR_DATABASE_MISMATCH;

 if( 0 != strcmp( pLogin->szUser, gszUser ))
  return ERR_USER_MISMATCH;

 if( 0 != strcmp( pLogin->szPassword, gszPassword ))
  return ERR_PASSWORD_MISMATCH;

 return ERR_DB_SUCCESS;
}


int
TPCCDisconnect( pCallersContext pCC )
{
 return ERR_DB_SUCCESS;
}



int
TPCCShutdown( void )
{
 int                                    retcode;

 retcode = ACMSxpShutdown( );
 if( ERR_SUCCESS != retcode )
  return retcode;

 retcode = DELIClientShutdown( );
 if( ERR_SUCCESS != retcode )
  return retcode;

 retcode = DELIServerShutdown( );
 if( ERR_SUCCESS != retcode )
  return retcode;

 return retcode;
}


void __cdecl
force_connect( void *arglist )
{
 LoginData              login;
 int                                    txnType;

 login.w_id = 0;
 login.ld_id = 0;
 login.pCC = 0;
 login.szApplication[0] = '\0';
 strcpy( login.szServer, gszServer );
 strcpy( login.szDatabase, gszDatabase );
 strcpy( login.szUser, gszUser );
 strcpy( login.szPassword, gszPassword );

 txnType = (int) arglist;
 switch ( txnType ) {
 case TYPE_DY:
  dy_transaction_init( STDL_SYNCHRONOUS, &login,
                                        (struct io_login_wksp *)&login );
  break;

 case TYPE_NO:
  no_transaction_init( STDL_SYNCHRONOUS, &login,
                                        (struct io_login_wksp *)&login );
  break;

 case TYPE_OS:
  os_transaction_init( STDL_SYNCHRONOUS, &login,
                                        (struct io_login_wksp *)&login );
  break;

 case TYPE_PT:
  pt_transaction_init( STDL_SYNCHRONOUS, &login,
                                        (struct io_login_wksp *)&login );
  break;

 case TYPE_SL:
  sl_transaction_init( STDL_SYNCHRONOUS, &login,
                                        (struct io_login_wksp *)&login );
  break;

 case TYPE_GC:
  gc_transaction_init( STDL_SYNCHRONOUS, &login,
                                        (struct io_login_wksp *)&login );
  break;
 }
 if ( login.status != ERR_DB_SUCCESS) {

  if ( ERR_DB_SUCCESS == gInitRetStatus )
   gInitRetStatus = ERR_FORCE_CONNECT_THREAD_FAILED;

  TPCCErr( "Connect Transaction returned %8X\r\n", login.status );
 }
 if ( InterlockedDecrement( &gForceAllThreadsCtr ) == 0 )
  SetEvent( gForceAllThreadsEvent );
 return;
}
```

tpcc_proc.sh

```
#################################################################
################
#
# tpcc_proc_case.sh
#
#################################################################
################
#
# This is the version of procs which was used in the Compaq-Sybase 11.G
# TPC-C benchmark (with the last-minute fixes) - March 26 1997
#
# This case script has the following changes from tpcc_proc_spec.sh
#   In new_order (both local and remote), the stock-item cursor, c_no_is
#   has been removed and replaced with an update-set-local variable stmt.
# Also CASE statements replace the nested if's.
#
# Also modified delivery proc where the ol and order table cursors have
#   been replaced by update_set_local_variable statements.
#
# In Payment procs, the cursor on customer, c_pay_c has been removed. Instead.
# added two update statements (with set local variables).
#
# Reinstated c_find cursor to find cust_id given c_last;
# Stock_level is back o its "single query" state!
#
#
#################################################################
################
#
#!/bin/sh -f

# Stored procedure for TPC-C 3.2 on SQL Server 11.1 and later
# Copyright Sybase 1997
#
isql -Usa -P <<EOF
use tpcc
go
if exists ( SELECT name FROM sysobjects WHERE name = 'neworder_local')
        DROP PROC neworder_local
go

CREATE PROC neworder_local (
        @w_id           smallint,
        @d_id           tinyint,
        @c_id           int,
        @o_ol_cnt       tinyint,

        @i_id       int = 0, @ol_qty        tinyint = 0,
        @i_id2      int = 0, @ol_qty2       tinyint = 0,
        @i_id3      int = 0, @ol_qty3       tinyint = 0,
        @i_id4      int = 0, @ol_qty4       tinyint = 0,
        @i_id5      int = 0, @ol_qty5       tinyint = 0,
        @i_id6      int = 0, @ol_qty6       tinyint = 0,
        @i_id7      int = 0, @ol_qty7       tinyint = 0,
        @i_id8      int = 0, @ol_qty8       tinyint = 0,
        @i_id9      int = 0, @ol_qty9       tinyint = 0,
        @i_id10     int = 0, @ol_qty10      tinyint = 0,
        @i_id11     int = 0, @ol_qty11      tinyint = 0,
        @i_id12     int = 0, @ol_qty12      tinyint = 0,
        @i_id13     int = 0, @ol_qty13      tinyint = 0,
        @i_id14     int = 0, @ol_qty14      tinyint = 0,
        @i_id15     int = 0, @ol_qty15      tinyint = 0
)
as

declare
        @w_tax          real,           @d_tax
        real,
        @c_last         char(16),   @c_credit   char(2),
        @c_discount     real,           @commit_flag
        tinyint,
        @i_price    real,
        @i_name             char(24),   @i_data
        char(50),

        @s_quantity         smallint,
        @s_ytd              int,            @s_order_cnt
        int,
        @s_dist             char(24),   @s_data
        char(50),

        @ol_number          tinyint,    @o_id           int,
        @o_entry_ddatetime, @b_g            char(1)

declare c_no_wdc CURSOR FOR
        SELECT  w_tax, d_tax, d_next_o_id,
                c_last, c_discount, c_credit
                ,1,0
                ,getdate()
                FROM    district HOLDLOCK,
                        warehouse HOLDLOCK,
                        customer (index c_clu prefetch 2 lru)
HOLDLOCK
                WHERE   d_w_id  = @w_id
                AND     d_id    = @d_id
                AND     w_id    = d_w_id
                AND     c_w_id  = d_w_id
                AND     c_d_id  = d_id
                AND     c_id    = @c_id
                FOR UPDATE OF d_next_o_id

begin

        begin transaction NO

        OPEN c_no_wdc
        FETCH c_no_wdc INTO
                @w_tax, @d_tax, @o_id,
                @c_last, @c_discount, @c_credit
                ,@commit_flag, @ol_number
                ,@o_entry_d
        UPDATE  district
                SET     d_next_o_id = @o_id + 1
                WHERE CURRENT OF c_no_wdc
        CLOSE c_no_wdc

        while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + 1
                ,@i_id = case @ol_number
         when 1 then @i_id2
              when 2 then @i_id3
              when 3 then @i_id4
              when 4 then @i_id5
              when 5 then @i_id6
              when 6 then @i_id7
              when 7 then @i_id8
              when 8 then @i_id9
              when 9 then @i_id10
              when 10 then @i_id11
              when 11 then @i_id12
              when 12 then @i_id13
              when 13 then @i_id14
              when 14 then @i_id15
              else @i_id
              end
                      , @ol_qty = case @ol_number
              when 1 then @ol_qty2
              when 2 then @ol_qty3
              when 3 then @ol_qty4
              when 4 then @ol_qty5
              when 5 then @ol_qty6
              when 6 then @ol_qty7
              when 7 then @ol_qty8
```

```
                when 8 then @ol_qty9
                when 9 then @ol_qty10
                when 10 then @ol_qty11
                when 11 then @ol_qty12
                when 12 then @ol_qty13
                when 13 then @ol_qty14
                when 14 then @ol_qty15
                else @ol_qty
                end

    /* set i_id, ol_qty for this lineitem */
    /* this is replaced by case statement */

    /* convert c_no_is cursor to a simple select */
    /* get item data (no one update item) */

                select @i_price = i_price,
                    @i_name = i_name,
                    @i_data = i_data
                  from item HOLDLOCK
                        where i_id = @i_id

                if (@@rowcount = 0)
                  begin
                   select @commit_flag = 0
                   select NULL, NULL, NULL, NULL
                   continue
                  end
      /*Otherwise if the item is found */
                  update stock
                    set s_ytd = s_ytd + @ol_qty,
                            @s_quantity = s_quantity - @ol_qty +
                        case when (s_quantity - @ol_qty < 10)
                        then 91 else 0 end,
                      s_quantity = s_quantity - @ol_qty +
                        case when (s_quantity - @ol_qty < 10)
                        then 91 else 0 end,
                            s_order_cnt = s_order_cnt + 1,
                        @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_w_id = @w_id and
                                    s_i_id = @i_id
                 if (@@rowcount = 0)
                   begin
                    select @commit_flag = 0
                    select NULL, NULL, NULL, NULL
                    continue
                            end
                  /*Otherwise if the Stock is found */
/* Compaq NT loader used Jan 01 1800 as NULL date */
              INSERT INTO order_line (
                 ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                 ol_supply_w_id, ol_delivery_d, ol_quantity,
                 ol_amount, ol_dist_info)
              VALUES (
                 @o_id, @d_id, @w_id, @ol_number, @i_id,
                 @w_id, "18000101", @ol_qty,
                 @ol_qty * @i_price, @s_dist)


      /* send line-item data to client */

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

                end /* while */

                INSERT INTO orders (
                            o_id, o_c_id, o_d_id, o_w_id,
                            o_entry_d, o_carrier_id, o_ol_cnt, o_all_local)
                VALUES (
                            @o_id, @c_id, @d_id, @w_id,
                            @o_entry_d, -1, @o_ol_cnt, 1)
                INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
                VALUES (@o_id, @d_id, @w_id)

                if (@commit_flag = 1)
                            commit transaction NO
                else
                            rollback transaction NO

                select                  /* Return to client */
                            @w_tax, @d_tax, @o_id, @c_last,
                            @c_discount, @c_credit, @o_entry_d
end
go

if exists ( SELECT name FROM sysobjects WHERE name = 'neworder_remote')
                DROP PROC neworder_remote
go
CREATE PROC neworder_remote (
                @w_id               smallint,
                @d_id               tinyint,
                @c_id               int,
                @o_ol_cnt  tinyint,

                @i_id       int = 0, @s_w_id        smallint = 0, @ol_qty
                tinyint = 0,
                @i_id2      int = 0, @s_w_id2       smallint = 0, @ol_qty2
                tinyint = 0,
                @i_id3      int = 0, @s_w_id3       smallint = 0, @ol_qty3
                tinyint = 0,
                @i_id4      int = 0, @s_w_id4       smallint = 0, @ol_qty4
                tinyint = 0,
                @i_id5      int = 0, @s_w_id5       smallint = 0, @ol_qty5
                tinyint = 0,
                @i_id6      int = 0, @s_w_id6       smallint = 0, @ol_qty6
                tinyint = 0,
                @i_id7      int = 0, @s_w_id7       smallint = 0, @ol_qty7
                tinyint = 0,
                @i_id8      int = 0, @s_w_id8       smallint = 0, @ol_qty8
                tinyint = 0,
                @i_id9      int = 0, @s_w_id9       smallint = 0, @ol_qty9
                tinyint = 0,
                @i_id10     int = 0, @s_w_id10      smallint = 0, @ol_qty10
                tinyint = 0,
                @i_id11     int = 0, @s_w_id11      smallint = 0, @ol_qty11
                tinyint = 0,
                @i_id12     int = 0, @s_w_id12      smallint = 0, @ol_qty12
                tinyint = 0,
                @i_id13     int = 0, @s_w_id13      smallint = 0, @ol_qty13
                tinyint = 0,
                @i_id14     int = 0, @s_w_id14      smallint = 0, @ol_qty14
                tinyint = 0,
                @i_id15     int = 0, @s_w_id15      smallint = 0, @ol_qty15
                tinyint = 0
)
as
```

```
declare
        @w_tax          real,           @d_tax
        real,
        @c_last         char(16),       @c_credit   char(2),
        @c_discount     real,           @commit_flag
        tinyint,

        @i_price    real,
        @i_name         char(24),       @i_data
        char(50),

        @s_quantity     smallint,
        @s_ytd          int,            @s_order_cnt
        int,
        @s_dist         char(24),       @s_data
        char(50),
        @s_remote_cnt   int,            @remote
        int,

        @ol_number      tinyint,        @o_id           int,
        @o_entry_ddatetime,     @b_g            char(1)

declare c_no_wdc CURSOR FOR
        SELECT  w_tax, d_tax, d_next_o_id,
                c_last, c_discount, c_credit
                ,1,0
                ,getdate()
                FROM    district HOLDLOCK,
                        warehouse HOLDLOCK,
                        customer (index c_clu prefetch 2 lru)
HOLDLOCK
                WHERE   d_w_id      = @w_id
                AND     d_id        = @d_id
                AND     w_id        = d_w_id
                AND     c_w_id      = d_w_id
                AND     c_d_id      = d_id
                AND     c_id        = @c_id
                FOR UPDATE OF d_next_o_id

begin

        begin transaction NO

        OPEN c_no_wdc
        FETCH c_no_wdc INTO
                @w_tax, @d_tax, @o_id,
                @c_last, @c_discount, @c_credit
                ,@commit_flag, @ol_number
                ,@o_entry_d
        UPDATE  district
                SET     d_next_o_id = @o_id + 1
                WHERE CURRENT OF c_no_wdc
        CLOSE c_no_wdc

        while (@ol_number < @o_ol_cnt) begin
                SELECT @ol_number = @ol_number + 1
                ,@i_id = case @ol_number
         when 1 then @i_id2
              when 2 then @i_id3
              when 3 then @i_id4
              when 4 then @i_id5
              when 5 then @i_id6
              when 6 then @i_id7
              when 7 then @i_id8
              when 8 then @i_id9
              when 9 then @i_id10
              when 10 then @i_id11
              when 11 then @i_id12
              when 12 then @i_id13
              when 13 then @i_id14
              when 14 then @i_id15
              else @i_id
                end
                        , @ol_qty = case @ol_number
              when 1 then @ol_qty2
              when 2 then @ol_qty3
              when 3 then @ol_qty4
              when 4 then @ol_qty5
              when 5 then @ol_qty6
              when 6 then @ol_qty7
              when 7 then @ol_qty8
              when 8 then @ol_qty9
              when 9 then @ol_qty10
              when 10 then @ol_qty11
              when 11 then @ol_qty12
              when 12 then @ol_qty13
              when 13 then @ol_qty14
              when 14 then @ol_qty15
              else @ol_qty
              end
                        ,@s_w_id = case @ol_number
            when 1 then @s_w_id2
              when 2 then @s_w_id3
              when 3 then @s_w_id4
              when 4 then @s_w_id5
              when 5 then @s_w_id6
              when 6 then @s_w_id7
              when 7 then @s_w_id8
              when 8 then @s_w_id9
              when 9 then @s_w_id10
              when 10 then @s_w_id11
              when 11 then @s_w_id12
              when 12 then @s_w_id13
              when 13 then @s_w_id14
              when 14 then @s_w_id15
              else @s_w_id
              end

            /* convert c_no_is cursor to a simple select */
            /* get item data (no one update item) */
                        select @i_price = i_price,
                            @i_name = i_name ,
                            @i_data = i_data
                          from item HOLDLOCK
                          where i_id = @i_id

                        if (@@rowcount = 0)
                          begin
                              select @commit_flag = 0
                              select NULL, NULL, NULL, NULL
                                      continue
                          end
             /* Otherwise if the item is found */
               update stock
                 set s_ytd = s_ytd + @ol_qty,
                   @s_quantity = s_quantity - @ol_qty +
                            case when (s_quantity - @ol_qty < 10)
                            then 91 else 0 end,
                      s_quantity = s_quantity - @ol_qty +
            case when (s_quantity - @ol_qty < 10)
             then 91 else 0 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,
```

```
                        s_order_cnt = s_order_cnt + 1,
                        s_remote_cnt = s_remote_cnt +
                        case when (@s_w_id = @w_id)
                then 0 else 1 end
        where s_w_id = @w_id and
            s_i_id = @i_id

        if (@@rowcount = 0)
          begin
            select @commit_flag = 0
            select NULL, NULL, NULL, NULL
            continue
          end
/* Compaq NT loader used Jan 01 1800 as NULL date */
              INSERT INTO order_line (
                  ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
                  ol_supply_w_id, ol_delivery_d, ol_quantity,
                  ol_amount, ol_dist_info)
              VALUES (
                  @o_id, @d_id, @w_id, @ol_number, @i_id,
                  @w_id, "18000101", @ol_qty,
                  @ol_qty * @i_price, @s_dist)

                  /* send line-item to client */
                  select
                        @i_name,
        @s_quantity,
        @i_price,
        b_g = case when ((patindex("%ORIGINAL%", @i_data) > 0) and
                (patindex("%ORIGINAL%", @s_data) > 0))
            then "B" else "G" end
            end /* while */

            INSERT INTO orders (
                        o_id, o_c_id, o_d_id, o_w_id,
                        o_entry_d, o_carrier_id, o_ol_cnt, o_all_local)
            VALUES (
                        @o_id, @c_id, @d_id, @w_id,
                        @o_entry_d, -1, @o_ol_cnt, 0)
            INSERT INTO new_order (no_o_id, no_d_id, no_w_id)
            VALUES (@o_id, @d_id, @w_id)

            if (@commit_flag = 1)
                        commit transaction NO
            else
                        rollback transaction NO

            select                  /* Return to client */
                        @w_tax, @d_tax, @o_id, @c_last,
                        @c_discount, @c_credit, @o_entry_d
end
go
if exists (select * from sysobjects where name = 'payment_byid')
            DROP PROC payment_byid
go
CREATE PROC payment_byid
            @w_id                   smallint,   @c_w_id
            smallint,
            @h_amount               float,
            @d_id                   tinyint,    @c_d_id
            tinyint,
            @c_id       int
as
declare     @c_last                 char(16)

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),
            @w_ytd                  float

declare     @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),
            @d_ytd                  float

declare     @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(12,0),          @c_balance
            float,
            @c_discount             real,
            @data1                  char(250),  @data2
            char(250),
            @c_data_1   char(250),  @c_data_2   char(250)

declare @screen_data    char(200),  @today      datetime

declare c_pay_wd CURSOR FOR
            SELECT      w_street_1, w_street_2, w_city,
                        w_state, w_zip, w_name, w_ytd,
                        d_street_1, d_street_2, d_city,
                        d_state, d_zip, d_name, d_ytd
            FROM        district HOLDLOCK,
                        warehouse HOLDLOCK
            WHERE       d_w_id      = @w_id
            AND         d_id        = @d_id
            AND         w_id        = d_w_id
            FOR UPDATE OF w_ytd, d_ytd

BEGIN TRANSACTION PID
            OPEN c_pay_wd
            FETCH c_pay_wd INTO
                        @w_street_1, @w_street_2, @w_city,
                        @w_state, @w_zip, @w_name, @w_ytd,
                        @d_street_1, @d_street_2, @d_city,
                        @d_state, @d_zip, @d_name, @d_ytd
            UPDATE district
                        SET     d_ytd  = @d_ytd + @h_amount
                        WHERE CURRENT OF c_pay_wd
            UPDATE warehouse
                        SET w_ytd = @w_ytd + @h_amount
                        WHERE CURRENT OF c_pay_wd
            CLOSE c_pay_wd

            /* Customer data */
            UPDATE customer SET
                         @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_balance = c_balance - @h_amount
                        , @c_balance = c_balance - @h_amount
                        , c_ytd_payment = c_ytd_payment + @h_amount
                        , c_payment_cnt = c_payment_cnt + 1
                        , @c_since = c_since
                        , @data1 = c_data1
                        , @data2 = c_data2
                        , @today = getdate()
            where
                        c_id = @c_id
```

```
                and c_w_id = @c_w_id
                and c_d_id = @c_d_id

        if (@c_credit = "BC")
        begin
                SELECT  @c_data_2 =
                        substring(@data1, 209, 42) +
                        substring(@data2, 1, 208)
                        ,@c_data_1 =
                        convert(char(5), @c_id) +
                        convert(char(4), @c_d_id) +
                        convert(char(5), @c_w_id) +
                        convert(char(4), @d_id) +
                        convert(char(5), @w_id) +
                        convert(char(19), @h_amount/100) +
substring(@data1, 1, 208)

                UPDATE customer SET
                          c_data1 = @c_data_1
                        , c_data2 = @c_data_2
                        , @screen_data = substring(@c_data_1, 1,
200)
                WHERE
                        c_id = @c_id
                        AND c_w_id = @c_w_id
                        AND c_d_id = @c_d_id
        end /* if */

        /* Create the history record */
        INSERT INTO history (
                h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                h_date, h_amount, h_data)
        VALUES (
                @c_id, @c_d_id, @c_w_id, @d_id, @w_id,
                @today, @h_amount, (@w_name + "    " + @d_name))

COMMIT TRANSACTION PID

        select                  /* Return to client */
                @c_id,
                @c_last,
                @today,
                @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
if exists (select * from sysobjects where name = 'payment_byname')
        DROP PROC payment_byname
go
```

```
CREATE PROC payment_byname
        @w_id                   smallint,       @c_w_id
        smallint,
        @h_amount               float,
        @d_id                   tinyint,        @c_d_id
        tinyint,
        @c_last     char(16)
as
declare     @n          int,        @c_id       int

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),
        @w_ytd                  float

declare     @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),
        @d_ytd                  float

declare     @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(12,0),              @c_balance
        float,
        @c_discount             real,
        @data1                  char(250),      @data2
        char(250),
        @c_data_1   char(250),  @c_data_2   char(250)

declare @screen_data    char(200),  @today      datetime

declare c_pay_wd CURSOR FOR
        SELECT      w_street_1, w_street_2, w_city,
                    w_state, w_zip, w_name, w_ytd,
                    d_street_1, d_street_2, d_city,
                    d_state, d_zip, d_name, d_ytd
        FROM        district HOLDLOCK,
                    warehouse HOLDLOCK
        WHERE       d_w_id      = @w_id
        AND         d_id        = @d_id
        AND         w_id        = d_w_id
        FOR UPDATE OF w_ytd, d_ytd

declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
        WHERE   c_w_id  =       @c_w_id
        AND     c_d_id  =       @c_d_id
        AND     c_last  =       @c_last
        ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
        FOR READ ONLY

BEGIN TRANSACTION PNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
        WHERE       c_w_id = @c_w_id and
                    c_d_id = @c_d_id and
                    c_last = @c_last
        OPEN c_find
        while (@n>0) begin
                    FETCH c_find INTO @c_id
                    SELECT @n = @n-1
        end
        CLOSE c_find
```

```
        OPEN c_pay_wd
        FETCH c_pay_wd INTO
                @w_street_1, @w_street_2, @w_city,
                @w_state, @w_zip, @w_name, @w_ytd,
                @d_street_1, @d_street_2, @d_city,
                @d_state, @d_zip, @d_name, @d_ytd
        UPDATE district
                SET     d_ytd  = @d_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        UPDATE warehouse
                SET w_ytd = @w_ytd + @h_amount
                WHERE CURRENT OF c_pay_wd
        CLOSE c_pay_wd

        /* Customer data */
        UPDATE customer SET
                 @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_balance = c_balance - @h_amount
                , @c_balance = c_balance - @h_amount
                , c_ytd_payment = c_ytd_payment + @h_amount
                , c_payment_cnt = c_payment_cnt + 1
                , @c_since = c_since
                , @data1 = c_data1
                , @data2 = c_data2
                , @today = getdate()
        where
                c_id = @c_id
                and c_w_id = @c_w_id
                and c_d_id = @c_d_id

        SELECT     @screen_data = NULL
        if (@c_credit = "BC")
        begin
                SELECT  @c_data_2 =
                        substring(@data1, 209, 42) +
                        substring(@data2, 1, 208)
                        ,@c_data_1 =
                        convert(char(5), @c_id) +
                        convert(char(4), @c_d_id) +
                        convert(char(5), @c_w_id) +
                        convert(char(4), @d_id) +
                        convert(char(5), @w_id) +
                        convert(char(19), @h_amount/100) +
substring(@data1, 1, 208)

                UPDATE customer SET
                         c_data1 = @c_data_1
                        , c_data2 = @c_data_2
                        , @screen_data = substring(@c_data_1, 1,
200)
                WHERE
                        c_id = @c_id
                        AND c_w_id = @c_w_id
                        AND c_d_id = @c_d_id
        end /* if */

        /* Create the history record */
        INSERT INTO history (
                h_c_id, h_c_d_id, h_c_w_id, h_d_id, h_w_id,
                h_date, h_amount, h_data)
        VALUES (
                @c_id, @c_d_id, @c_w_id, @d_id, @w_id,
                @today, @h_amount, (@w_name + "    " + @d_name))

COMMIT TRANSACTION PNM

        select                  /* Return to client */
                @c_id,
                @c_last,
                @today,
                @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
if exists (select * from sysobjects where name = 'order_status_byid')
        DROP PROC order_status_byid
go
CREATE PROC order_status_byid
        @w_id           smallint,
        @d_id           tinyint,
        @c_id           int
as

DECLARE  @o_id          int,
        @o_entry_ddatetime,
        @o_carrier_id   smallint

BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
        SELECT  @o_id = o_id, @o_carrier_id = o_carrier_id,
                @o_entry_d = o_entry_d
        FROM    orders (index o_clu prefetch 16 mru) HOLDLOCK
        WHERE   o_w_id  = @w_id
        AND     o_d_id  = @d_id
        AND     o_c_id  = @c_id
        /* ORDER BY o_w_id, o_d_id, o_id */

/* Select order lines for the current order */
        select  /* Return multiple rows to client */
                ol_supply_w_id,
                ol_i_id,
                ol_quantity,
                ol_amount,
                ol_delivery_d
        FROM    order_line HOLDLOCK
        WHERE   ol_o_id = @o_id
        AND     ol_d_id = @d_id
        AND     ol_w_id = @w_id
```

```
        select      /* Return single row to client */
                    @c_id, c_last, c_first, c_middle, c_balance,
                    @o_id,
                    @o_entry_d,
                    @o_carrier_id
        FROM        customer (index c_clu prefetch 2 lru) HOLDLOCK
        WHERE   c_id    = @c_id
        AND         c_d_id  = @d_id
        AND         c_w_id  = @w_id

COMMIT TRANSACTION OSID
go
if exists (select * from sysobjects where name = 'order_status_byname')
        DROP PROC order_status_byname
go
CREATE PROC order_status_byname
        @w_id                   smallint,
        @d_id                   tinyint,
        @c_last     char(16)
as

DECLARE  @o_id                  int,
        @o_entry_d  datetime,
        @o_carrier_id           smallint

declare     @n          int,        @c_id       int
declare c_find CURSOR FOR
        SELECT  c_id
        FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
        WHERE   c_w_id  =     @w_id
        AND     c_d_id  =     @d_id
        AND     c_last  =     @c_last
        ORDER BY c_w_id, c_d_id, c_last, c_first, c_id
        FOR READ ONLY

BEGIN TRANSACTION OSNM
        SELECT @n = (count(*)+1)/2
        FROM customer (index c_non1 prefetch 2 lru) HOLDLOCK
        WHERE       c_w_id = @w_id and
                    c_d_id = @d_id and
                    c_last = @c_last
        OPEN c_find
        while (@n>0) begin
                    FETCH c_find INTO @c_id
                    SELECT @n = @n-1
        end
        CLOSE c_find

/* Get the latest order made by the customer */
        SELECT      @o_id = o_id, @o_carrier_id = o_carrier_id,
                    @o_entry_d = o_entry_d
        FROM        orders (index o_clu prefetch 16 mru) HOLDLOCK
        WHERE       o_w_id      = @w_id
        AND         o_d_id      = @d_id
        AND         o_c_id      = @c_id
        /* ORDER BY o_w_id, o_d_id, o_id */

/* Select order lines for the current order */
        select      /* Return multiple rows to client */
                    ol_supply_w_id,
                    ol_i_id,
                    ol_quantity,
                    ol_amount,
                    ol_delivery_d
        FROM        order_line HOLDLOCK
        WHERE       ol_o_id = @o_id
        AND         ol_d_id = @d_id
        AND         ol_w_id = @w_id

        select      /* Return single row to client */
                    @c_id, c_last, c_first, c_middle, c_balance,
                    @o_id,
                    @o_entry_d,
                    @o_carrier_id
        FROM        customer (index c_clu prefetch 2 lru) HOLDLOCK
        WHERE   c_id    = @c_id
        AND         c_d_id  = @d_id
        AND         c_w_id  = @w_id

COMMIT TRANSACTION OSNM
go
if exists (select * from sysobjects where name = 'delivery')
        drop proc delivery
go
CREATE PROC delivery
        @w_id                   smallint,
        @o_carrier_id           smallint,
        @d_id                   tinyint = 1
as

declare     @no_o_id    int,                    @o_c_id
            smallint,
            @ol_total   float,                  @ol_amount
            float,
            @junk_id    smallint

declare c_del_no CURSOR FOR
        SELECT      no_o_id
        FROM        new_order (index no_clu) HOLDLOCK
        WHERE       no_d_id = @d_id
        AND         no_w_id = @w_id
        FOR UPDATE
        /*
        ** The only purpose of the index hint in the above is to ensure
        ** that the clustered index is used.  As it turns out, our optimizer
        ** chooses the clustered index anyway -- with or without the hint.
        */

begin

   while (@d_id <= 10) begin

        BEGIN TRANSACTION DEL
        OPEN c_del_no
        FETCH c_del_no INTO @no_o_id

        if (@@sqlstatus != 0)
        begin
                    COMMIT TRANSACTION DEL
                    select NULL
                    CLOSE c_del_no
                    select @d_id = @d_id + 1
                    continue
        end

        DELETE      FROM new_order
        WHERE       CURRENT OF c_del_no
        CLOSE       c_del_no

/* Using the 'update' enhancement */
        UPDATE      orders
        SET         o_carrier_id = @o_carrier_id,
                    @o_c_id    = o_c_id,
                    @ol_total = 0.0
    WHERE           o_id        = @no_o_id
    AND     o_d_id      = @d_id
    AND     o_w_id      = @w_id

        UPDATE      order_line
        SET         ol_delivery_d = getdate(),
                    @ol_total = ol_amount + @ol_total
        WHERE       ol_o_id     = @no_o_id
        AND         ol_d_id     = @d_id
        AND         ol_w_id     = @w_id
```

```
            UPDATE    customer
            SET    c_balance          = c_balance + @ol_total,
             c_delivery_cnt           = c_delivery_cnt + 1
            WHERE  c_id               = @o_c_id
            AND    c_d_id             = @d_id
            AND    c_w_id             = @w_id

            COMMIT TRANSACTION DEL

            select            /* Return to client */
                                    @no_o_id
            select @d_id = @d_id + 1
   end /* while @d_id... */
end
go
if exists ( SELECT name FROM sysobjects WHERE name = 'stock_level')
   DROP PROC stock_level
go

CREATE PROC stock_level
            @w_id       smallint,
            @d_id       tinyint,
            @threshold smallint
as
            select  s_i_id
            FROM        district,
                        order_line (index ol_clu prefetch 2 lru),
                        stock (index s_clu prefetch 2 lru)
            WHERE       d_w_id      =           @w_id
            AND         d_id        =           @d_id
            AND         ol_w_id     =           @w_id
            AND         ol_d_id     =           @d_id
            AND         ol_o_id     between (d_next_o_id - 20) and
(d_next_o_id - 1)
            AND         s_w_id      =           ol_w_id
            AND         s_i_id      =           ol_i_id
            AND    s_quantity < @threshold
go
EOF
```

tpcc_ps.c

```
/*+**********************************************************************
 *                                                                     *
 * COPYRIGHT (c) 1997 BY                                               *
 * COMPAQ COMPUTER CORPORATION, .                                      *
 * ALL RIGHTS RESERVED.                                                *
 *                                                                     *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
 * ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
 * OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
 * TRANSFERRED.                                                        *
 *                                                                     *
 * THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
 * AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY COMPAQ COMPUTER  *
 * CORPORATION.                                                        *
 *                                                                     *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.              *
 *                                                                     *
 *                                                                     *
 **********************************************************************/

#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <sys/types.h>
#include <windows.h>
#include <process.h>
#include <Winsock.h>
#include <tpccstruct.h>
#include <tpccapi.h>
#include <tpccerr.h>

#include <tpcc_stdl_workspaces.h>
#include <tpcc_acmsxp.h>

#include <tpcc.h>

#ifdef FFE_DEBUG
# include <crtdbg.h>
#endif

static DWORD  gTLSDBContext = 0;

static LONG   gForceThreadStartCtr;
static HANDLE gForceThreadStartEvent;



void
transaction_init( char *type, pLoginData pLogin )
{
 char                                tmp[16];
 DWORD                               dwTID;

 DBContext DBC  = (DBContext) TlsGetValue(gTLSDBContext);

 if (DBC == INVALID_DB_CONTEXT) {

  gethostname(tmp, sizeof(tmp));
  dwTID = GetCurrentThreadId( );
  sprintf( pLogin->szApplication, "%s:%s - %d", tmp, type, dwTID );

  pLogin->status = TPCCConnectDB( &DBC, pLogin );

  TPCCLog( "%s, dbprocptr = %8X\r\n", pLogin->szApplication, DBC );

  if ( pLogin->status == ERR_DB_SUCCESS ) {
   if (TlsSetValue(gTLSDBContext, DBC) == 0)
            pLogin->status =
ERR_CANT_SET_THREAD_LOCAL_STORAGE;
  }

  if ( InterlockedDecrement( &gForceThreadStartCtr ) == 0 )
   SetEvent( gForceThreadStartEvent );

  WaitForSingleObject( gForceThreadStartEvent, INFINITE );
 }
 else {
  pLogin->status = ERR_ALREADY_LOGGED_IN;
 }
}


void
dy_transaction_init( struct io_login_wksp *login_wksp )
{
 transaction_init( "DY", (pLoginData)login_wksp );
}


void
no_transaction_init( struct io_login_wksp *login_wksp )
{
 transaction_init( "NO", (pLoginData)login_wksp );
}


void
os_transaction_init( struct io_login_wksp *login_wksp )
{
```

```
 transaction_init( "OS", (pLoginData)login_wksp );
}


void
pt_transaction_init( struct io_login_wksp *login_wksp )
{
 transaction_init( "PT", (pLoginData)login_wksp );
}


void
sl_transaction_init( struct io_login_wksp *login_wksp )
{
 transaction_init( "SL", (pLoginData)login_wksp );
}


void
gc_transaction_init( struct io_login_wksp *login_wksp )
{
 transaction_init( "GC", (pLoginData)login_wksp );
}


void
dy_transaction( struct io_dy_wksp *dy_wksp )
{
 DBContext DBC  = (DBContext) TlsGetValue(gTLSDBContext);
 pDeliveryData ptr;

 ptr = (pDeliveryData)dy_wksp;

#ifdef FFE_DEBUG
   ptr->iStage |= IN_RH;
   ptr->dwXPThreadId = GetCurrentThreadId( );
#endif

 if (DBC == INVALID_DB_CONTEXT) {
   ptr->status = ERR_DB_NOT_LOGGED_IN;
 }
 else {
   ptr->status = TPCCDeliveryDB( DBC, ptr );
 }

#ifdef FFE_DEBUG
   _ASSERT( VALID_DB_ERR(ptr->status) );
   ptr->iStage |= LEAVING_RH;
#endif
}


void
no_transaction( struct io_no_wksp *no_wksp )
{
 DBContext DBC  = (DBContext) TlsGetValue(gTLSDBContext);
 pNewOrderData ptr;

 ptr = (pNewOrderData)no_wksp;

#ifdef FFE_DEBUG
   ptr->iStage |= IN_RH;
   ptr->dwXPThreadId = GetCurrentThreadId( );
#endif

 if (DBC == INVALID_DB_CONTEXT) {
   ptr->status = ERR_DB_NOT_LOGGED_IN;
 }
 else {
   ptr->status = TPCCNewOrderDB( DBC, ptr );
 }

#ifdef FFE_DEBUG
   _ASSERT( VALID_DB_ERR(ptr->status) );
   ptr->iStage |= LEAVING_RH;
#endif
}


void
os_transaction( struct io_os_wksp *os_wksp )
{
 DBContext DBC  = (DBContext) TlsGetValue(gTLSDBContext);
 pOrderStatusData ptr;

 ptr = (pOrderStatusData)os_wksp;

#ifdef FFE_DEBUG
   ptr->iStage |= IN_RH;
   ptr->dwXPThreadId = GetCurrentThreadId( );
#endif

 if (DBC == INVALID_DB_CONTEXT) {
   ptr->status = ERR_DB_NOT_LOGGED_IN;
 }
 else {
   ptr->status = TPCCOrderStatusDB( DBC, ptr );
 }

#ifdef FFE_DEBUG
   _ASSERT( VALID_DB_ERR(ptr->status) );
   ptr->iStage |= LEAVING_RH;
#endif
}


void
pt_transaction( struct io_pt_wksp *pt_wksp )
{
 DBContext DBC  = (DBContext) TlsGetValue(gTLSDBContext);
 pPaymentData ptr;

 ptr = (pPaymentData)pt_wksp;

#ifdef FFE_DEBUG
   ptr->iStage |= IN_RH;
   ptr->dwXPThreadId = GetCurrentThreadId( );
#endif

 if (DBC == INVALID_DB_CONTEXT) {
   ptr->status = ERR_DB_NOT_LOGGED_IN;
 }
 else {
   ptr->status = TPCCPaymentDB( DBC, ptr );
 }

#ifdef FFE_DEBUG
   _ASSERT( VALID_DB_ERR(ptr->status) );
   ptr->iStage |= LEAVING_RH;
#endif
}


void
sl_transaction( struct io_sl_wksp *sl_wksp )
{
 DBContext DBC  = (DBContext) TlsGetValue(gTLSDBContext);
 pStockLevelData ptr;

 ptr = (pStockLevelData)sl_wksp;

#ifdef FFE_DEBUG
   ptr->iStage |= IN_RH;
   ptr->dwXPThreadId = GetCurrentThreadId( );
```

```
#endif

 if (DBC == INVALID_DB_CONTEXT) {
  ptr->status = ERR_DB_NOT_LOGGED_IN;
 }
 else {
  ptr->status = TPCCStockLevelDB( DBC, ptr );
 }

#ifdef FFE_DEBUG
  _ASSERT( VALID_DB_ERR(ptr->status) );
  ptr->iStage |= LEAVING_RH;
#endif
}


void
gc_transaction( struct io_gc_wksp *gc_wksp, struct int_gc_wksp *ptype )
{
 int transaction_type;

 transaction_type = ptype->trans;
 switch( transaction_type ) {
 case TYPE_DY:
  dy_transaction( (struct io_dy_wksp *)gc_wksp );
  break;
 case TYPE_NO:
  no_transaction( (struct io_no_wksp *)gc_wksp );
  break;
 case TYPE_OS:
  os_transaction( (struct io_os_wksp *)gc_wksp );
  break;
 case TYPE_PT:
  pt_transaction( (struct io_pt_wksp *)gc_wksp );
  break;
 case TYPE_SL:
  sl_transaction( (struct io_sl_wksp *)gc_wksp );
  break;
 default:
#ifdef FFE_DEBUG
  _ASSERT( FALSE );
#else
  ;
#endif
 }
}


int ACMSxpStartup( int maxDBConnections, long totalGovValue )
{
int ret_status;


 if ((gTLSDBContext = TlsAlloc()) == 0xFFFFFFFF)
  return ERR_CANT_ALLOCATE_THREAD_LOCAL_STORAGE;



 gForceThreadStartCtr = totalGovValue;

 gForceThreadStartEvent = CreateEvent( NULL, TRUE, FALSE, NULL );
 if ( gForceThreadStartEvent == NULL )
  return ERR_CANT_CREATE_FORCE_THRED_STRT_EVENT;

 ret_status = TPCCStartupDB( maxDBConnections );
 return ret_status;
}


int ACMSxpShutdown( void )
{
 CloseHandle( gForceThreadStartEvent );
 TlsFree( gTLSDBContext );
 return( TPCCShutdownDB() );
}
```

tpccapi.h

```
#ifndef TPCCAPI_H
#define TPCCAPI_H
/*+*************************************************************************
 *                                                                         *
 * COPYRIGHT (c) 1996 BY                                                   *
 * COMPAQ COMPUTER CORPORATION, .                                          *
 * ALL RIGHTS RESERVED.                                                    *
 *                                                                         *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE          *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER     *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
 * TRANSFERRED.                                                            *
 *                                                                         *
 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE    *
 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY COMPAQ COMPUTER          *
 * CORPORATION.                                                            *
 *                                                                         *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                  *
 *                                                                         *
 *                                                                         *
 **************************************************************************/

/*+*************************************************************************
**************************** tpccapi.h ***********************************
***************************************************************************
*
** tpccapi.h:  This header file declares function calls between TPCC
**          application and server
*
*
*  Authors: Tareef Kawaf and Bill Carr
**
**
**  02-05-97 FWM Added bQueueDelivery flag to startup call.
**  18-Feb-98 WCarr                 Introduced TPCCAPI V2.0
**
*/


#define DELIVERY_RESPONSE_COUNT 2

int TPCCGetTransportData( pTransportData pTransport );

int TPCCStartup( int iMaxUsers, pTransportData pTransport );
int TPCCStartupDB( int iMaxDBConnections );

int TPCCConnect( pLoginData pLogin );
int TPCCConnectDB( DBContext *pDBC, pLoginData pLogin );

int TPCCDelivery( pDeliveryData pDelivery,
                  pDeliveryData
CompletedDeliveries[DELIVERY_RESPONSE_COUNT] );
int TPCCDeliveryDeferred( pDeliveryData pDelivery );
int TPCCDeliveryDB( DBContext DBC, pDeliveryData pDelivery );

int TPCCNewOrder( pNewOrderData pNewOrder );
int TPCCNewOrderDB( DBContext DBC, pNewOrderData pNewOrder );

int TPCCOrderStatus( pOrderStatusData pOrderStatus );
int TPCCOrderStatusDB( DBContext DBC, pOrderStatusData pOrderStatus );

int TPCCPayment( pPaymentData pPayment );
int TPCCPaymentDB( DBContext DBC, pPaymentData pPayment );

int TPCCStockLevel( pStockLevelData pStockLevel );
int TPCCStockLevelDB( DBContext DBC, pStockLevelData pStockLevel );

int TPCCCheckpoint( pCheckpointData pCheckpoint );
```

```
int TPCCCheckpointDB( DBContext DBC, pCheckpointData pCheckpoint );

int TPCCDisconnect( pCallersContext pCC );
int TPCCDisconnectDB( DBContext DBC, pCallersContext pCC );

int TPCCShutdown( void );
int TPCCShutdownDB( void );

void TPCCDeliveryResponse( int retcode, pDeliveryData pDelivery,
                           pDeliveryData
CompletedDeliveries[DELIVERY_RESPONSE_COUNT] );

void TPCCDeliveryDeferredResponse( int retcode, pDeliveryData pDelivery );

void TPCCNewOrderResponse( int retcode, pNewOrderData pNewOrder );

void TPCCOrderStatusResponse( int retcode, pOrderStatusData pOrderStatus );

void TPCCPaymentResponse( int retcode, pPaymentData pPayment );

void TPCCStockLevelResponse( int retcode, pStockLevelData pStockLevel );

void TPCCResponseComplete( CallersContext *pCC );

void ErrorMessage( CallersContext *pCC, int iError, int iErrorType,
                   char *pszMesasge );

int TPCCGetTransportErrorString( int iErrorCode, int iBufSize, char *pBuffer );
int TPCCGetDBErrorString( int iErrorCode, int iBufSize, char *pBuffer );

BOOL TPCCOpenLog( void );

BOOL TPCCCloseLog( void );

void TPCCLog( char *fmt, ... );

void TPCCErr( char *fmt, ... );

void TPCCTransactionErr( pConnData pConn, char *fmt, ... );
#endif
```

tpccerr.h

```
#ifndef TPCCERR_H
#define TPCCERR_H



#pragma message ("FIXME: the error types need to be made DB non-specific")
#define     ERR_TYPE_WEBDLL
                    1
#define     ERR_TYPE_SQL
                    2
#define     ERR_TYPE_DBLIB
                    3

#define ERR_DB_SUCCESS                          0
#define ERR_DB_ERROR                            1
#define ERR_TRANSPORT_ERROR                 2
#define ERR_DB_INTERFACE                    3
#define ERR_DB_DEADLOCK_LIMIT                   4
#define ERR_DB_NOT_COMMITED                     5
#define ERR_DB_DEAD                             6
#define ERR_DB_PENDING                          7
#define ERR_DB_NOT_LOGGED_IN                    8
#define ERR_DB_LOGIN_FAILED                 9
#define ERR_DB_USE_FAILED                   10
#define ERR_DB_LOGOUT_FAILED                    11

#define ERR_DB_MAX_ERR                          11

#define VALID_DB_ERR(err) (((err) >= ERR_DB_SUCCESS)&&((err) <=
ERR_DB_MAX_ERR))

#define     ERR_SUCCESS                             1000
#define     ERR_COMMAND_UNDEFINED                   1001
#define     ERR_NOT_IMPLEMENTED_YET                 1002
#define     ERR_CANNOT_INIT_TERMINAL            1003
#define     ERR_OUT_OF_MEMORY                       1004
#define     ERR_NEW_ORDER_NOT_PROCESSED             1005
#define     ERR_PAYMENT_NOT_PROCESSED               1006
#define     ERR_NO_SERVER_SPECIFIED                 1007
#define     ERR_ORDER_STATUS_NOT_PROCESSED          1008
#define     ERR_W_ID_INVALID                    1009
#define     ERR_CAN_NOT_SET_MAX_CONNECTIONS         1010
#define     ERR_NOSUCH_CUSTOMER                     1011
#define     ERR_D_ID_INVALID                    1012
#define     ERR_MAX_CONNECT_PARAM                   1013
#define     ERR_INVALID_SYNC_CONNECTION             1014
#define     ERR_INVALID_TERMID                      1015
#define     ERR_PAYMENT_INVALID_CUSTOMER            1016
#define     ERR_SQL_OPEN_CONNECTION                 1017
#define     ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY    1018
#define     ERR_STOCKLEVEL_THRESHOLD_INVALID    1019
#define     ERR_STOCKLEVEL_THRESHOLD_RANGE          1020
#define     ERR_STOCKLEVEL_NOT_PROCESSED            1021
#define     ERR_NEWORDER_FORM_MISSING_DID           1022
#define     ERR_NEWORDER_DISTRICT_INVALID           1023
#define     ERR_NEWORDER_DISTRICT_RANGE             1024
#define     ERR_NEWORDER_CUSTOMER_KEY               1025
#define     ERR_NEWORDER_CUSTOMER_INVALID           1026
#define     ERR_NEWORDER_CUSTOMER_RANGE             1027
#define     ERR_NEWORDER_MISSING_IID_KEY            1028
#define     ERR_NEWORDER_ITEM_BLANK_LINES           1029
#define     ERR_NEWORDER_ITEMID_INVALID             1030
#define     ERR_NEWORDER_MISSING_SUPPW_KEY          1031
#define     ERR_NEWORDER_SUPPW_INVALID              1032
#define     ERR_NEWORDER_MISSING_QTY_KEY            1033
#define     ERR_NEWORDER_QTY_INVALID            1034
#define     ERR_NEWORDER_SUPPW_RANGE                1035
#define     ERR_NEWORDER_ITEMID_RANGE               1036
#define     ERR_NEWORDER_QTY_RANGE                  1037
#define     ERR_PAYMENT_DISTRICT_INVALID            1038
#define     ERR_NEWORDER_SUPPW_WITHOUT_ITEMID       1039
#define     ERR_NEWORDER_QTY_WITHOUT_ITEMID         1040
#define     ERR_NEWORDER_NOITEMS_ENTERED            1041
#define     ERR_PAYMENT_MISSING_DID_KEY             1042
#define     ERR_PAYMENT_DISTRICT_RANGE              1043
#define     ERR_PAYMENT_MISSING_CID_KEY             1044
#define     ERR_PAYMENT_CUSTOMER_INVALID            1045
#define     ERR_PAYMENT_MISSING_CLT                 1046
#define     ERR_PAYMENT_LAST_NAME_TO_LONG           1047
#define     ERR_PAYMENT_CUSTOMER_RANGE              1048
#define     ERR_PAYMENT_CID_AND_CLT                 1049
#define     ERR_PAYMENT_MISSING_CDI_KEY             1050
#define     ERR_PAYMENT_CDI_INVALID                 1051
#define     ERR_PAYMENT_CDI_RANGE                   1052
#define     ERR_PAYMENT_MISSING_CWI_KEY             1053
#define     ERR_PAYMENT_CWI_INVALID                 1054
#define     ERR_PAYMENT_CWI_RANGE                   1055
#define     ERR_PAYMENT_MISSING_HAM_KEY             1056
#define     ERR_PAYMENT_HAM_INVALID                 1057
#define     ERR_PAYMENT_HAM_RANGE                   1058
#define     ERR_ORDERSTATUS_MISSING_DID_KEY         1059
#define     ERR_ORDERSTATUS_DID_INVALID             1060
#define     ERR_ORDERSTATUS_DID_RANGE               1061
#define     ERR_ORDERSTATUS_MISSING_CID_KEY         1062
#define     ERR_ORDERSTATUS_MISSING_CLT_KEY         1063
#define     ERR_ORDERSTATUS_CLT_RANGE               1064
#define     ERR_ORDERSTATUS_CID_INVALID             1065
#define     ERR_ORDERSTATUS_CID_RANGE               1066
#define     ERR_ORDERSTATUS_CID_AND_CLT             1067
```

```
#define       ERR_DELIVERY_MISSING_OCD_KEY                  1068
#define       ERR_DELIVERY_CARRIER_INVALID                  1069
#define       ERR_DELIVERY_CARRIER_ID_RANGE                 1070
#define ERR_PAYMENT_MISSING_CLT_KEY             1071
#define ERR_CANT_FIND_TPCC_KEY                  1072
#define ERR_CANT_FIND_INETINFO_KEY              1073
#define ERR_CANT_FIND_POOLTHREADLIMIT           1074
#define ERR_DB_DELIVERY_NOT_QUEUED              1075
#define ERR_DELIVERY_NOT_PROCESSED              1076
#define ERR_TERM_ALLOCATE_FAILED                1077
#define ERR_PENDING                             1078
#define ERR_CANT_START_FRCDINIT_THREAD          1079
#define ERR_CANT_START_DELIVERY_THREAD          1080
#define ERR_GOVERNOR_VALUE_NOT_FOUND      1081
#define ERR_SERVER_MISMATCH            1082
#define ERR_DATABASE_MISMATCH          1083
#define ERR_USER_MISMATCH              1084
#define ERR_PASSWORD_MISMATCH          1085
#define ERR_CANT_CREATE_ALL_THREADS_EVENT   1086
#define ERR_CANT_CREATE_FORCE_THRED_STRT_EVENT 1087
#define ERR_CANT_ALLOCATE_THREAD_LOCAL_STORAGE 1088
#define ERR_CANT_SET_THREAD_LOCAL_STORAGE   1089
#define ERR_FORCE_CONNECT_THREAD_FAILED         1090
#define ERR_CANT_FIND_SERVER_VALUE              1091
#define ERR_NO_MESSAGE                          1092
#define ERR_CANT_FIND_PATH_VALUE                1093
#define       ERR_CANNOT_CREATE_RESULTS_FILE            1094
#define ERR_DELIVERY_PIPE_SECURITY              1095
#define ERR_DELIVERY_PIPE_CREATE                1096
#define ERR_DELIVERY_PIPE_OPEN                  1097
#define ERR_DELIVERY_PIPE_READ                  1098
#define ERR_DELIVERY_PIPE_DISCONNECT            1099
#define ERR_CANT_FIND_DATABASE_VALUE            1100
#define ERR_CANT_FIND_USER_VALUE                1101
#define ERR_CANT_FIND_PASSWORD_VALUE            1102
#define ERR_DELIVERY_OUTPUT_PIPE_WRITE          1103
#define ERR_DELIVERY_OUTPUT_PIPE_READ           1104
#define ERR_DELIVERY_MISSING_QUEUETIME_KEY      1105
#define ERR_DELIVERY_QUEUETIME_INVALID          1106
#define ERR_ALREADY_LOGGED_IN                   1107
#define ERR_INVALID_FORM                        1109
#define ERR_DELIVERY_MUST_CONNECTDB             1110
#define ERR_INVALID_FORM_AND_CMD_NOT_BEGIN      1111
#define ERR_MAX_CONNECTIONS_EXCEEDED            1112
#define ERR_CANNOT_FIND_CONNECTION              1113
#define ERR_CKPT_NOT_INITIALIZED                1114


typedef struct _SERRORMSG
{
  int                   iError;
  char                  szMsg[80];
} SERRORMSG;


#ifdef TPCC_C
SERRORMSG errorMsgs[] =
{
  { ERR_SUCCESS, "Success, no error." },
  { ERR_NO_MESSAGE, "No message string available for the specified error
code." },
  { 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*\"." },
  { ERR_DB_ERROR, "A Database error has occurred." },
  { ERR_DELIVERY_NOT_PROCESSED, "Delivery not processed." },
  { ERR_DB_DELIVERY_NOT_QUEUED, "Delivery not queued." },
  { ERR_CANT_FIND_TPCC_KEY, "TPCC key not found in registry." },
  { ERR_CANT_FIND_INETINFO_KEY, "inetinfo key not found in registry." },
  { ERR_CANT_FIND_POOLTHREADLIMIT, "PoolThreadLimit value not set
in inetinfo\\Parameters key." },
  { ERR_TERM_ALLOCATE_FAILED, "Failed to allocate terminal data
structure." },
  { ERR_DELIVERY_PIPE_SECURITY, "Failed to initialize delivery pipe
security." },
  { ERR_DELIVERY_PIPE_CREATE, "Failed to create delivery pipe." },
  { ERR_DELIVERY_PIPE_OPEN, "Failed to open delivery pipe." },
  { ERR_DELIVERY_PIPE_READ, "Failed to read delivery pipe." },
  { ERR_DELIVERY_PIPE_DISCONNECT, "Failed to start delivery pipe
disconnect thread."},
  { ERR_PENDING, "Transaction pending."},
  { ERR_CANT_START_FRCDINIT_THREAD, "Can't start Forced
Initialization thread." },
  { ERR_CANT_START_DELIVERY_THREAD, "Can't start delivery thread."
},
  { ERR_GOVERNOR_VALUE_NOT_FOUND, "Governor value not found in
Registry." },
  { ERR_SERVER_MISMATCH, "Server does not match registry value." },
  { ERR_DATABASE_MISMATCH, "Database name does not match registry
value." },
  { ERR_USER_MISMATCH, "User name does not match registry value." },
  { ERR_PASSWORD_MISMATCH, "Password does not match registry value."
},
  { ERR_CANT_CREATE_ALL_THREADS_EVENT, "Can't create All Threads
Event." },
  { ERR_CANT_CREATE_FORCE_THRED_STRT_EVENT, "Can't create
Force Thread Start Event." },
  { ERR_CANT_ALLOCATE_THREAD_LOCAL_STORAGE, "Can't allocate
thread local storage" },
  { ERR_CANT_SET_THREAD_LOCAL_STORAGE, "Can't set thread local
storage." },
  { ERR_FORCE_CONNECT_THREAD_FAILED, "At least one database
connect call failed, check log files for specific error." },
  { ERR_CANT_FIND_SERVER_VALUE, "Server value not set in TPCC key."
},
  { ERR_CANT_FIND_PATH_VALUE, "PATH value not set in TPCC key." },
  { ERR_CANNOT_CREATE_RESULTS_FILE, "Cannot create results file." },
  { ERR_CANT_FIND_DATABASE_VALUE, "Database value not set in TPCC
key." },
  { ERR_CANT_FIND_USER_VALUE, "User value not set in TPCC key." },
  { ERR_CANT_FIND_PASSWORD_VALUE, "Password value not set in TPCC
key." },
  { ERR_DELIVERY_OUTPUT_PIPE_WRITE, "Failed to write output delivery
pipe." },
  { ERR_DELIVERY_OUTPUT_PIPE_READ, "Failed to read output delivery
pipe." },
  { ERR_DELIVERY_MISSING_QUEUETIME_KEY, "Delivery queue time
missing from query." },
  { ERR_DELIVERY_QUEUETIME_INVALID, "Delivery queue time is
invalid." },
  { ERR_ALREADY_LOGGED_IN, "TPCCConnectDB has already been
called." },
  { ERR_DB_NOT_LOGGED_IN, "TPCCConnectDB has not yet been called." },
  { ERR_INVALID_FORM, "The FORM field is missing or invalid." },
  { ERR_DELIVERY_MUST_CONNECTDB, "Synchronous transport requires
delivery server connect to database." },
  { ERR_INVALID_FORM_AND_CMD_NOT_BEGIN, "The FORM field is
missing and CMD is not Begin." },
  { ERR_MAX_CONNECTIONS_EXCEEDED, "The maximum number of
connections has been exceeded." },
  { ERR_CANNOT_FIND_CONNECTION, "Transport layer unable to find a
DBContext coresponding to the CallersContext." },
  { ERR_CKPT_NOT_INITIALIZED, "The checkpoint subsystem has not been
started." },
  {             0, "" }
};
#else
extern SERRORMSG errorMsgs[];
#endif

#endif
```

tpccstruct.h

```
#ifndef TPCCSTRUCT_H
#define TPCCSTRUCT_H




#include <time.h>

#ifdef _WIN32
# ifndef BOOLEAN
#  define BOOLEAN BOOL
# endif
#else
# include <sys/types.h>
```

```
# define BOOLEAN int
# define VMS     0
# define LINEMAX 256

# define FALSE   0
# define TRUE    1
#endif

#define MAX_OL  15


#ifdef FFE_DEBUG

# define CALLING_LH  0x0001
# define IN_LH                  0x0002
# define IN_RH                  0x0004
# define IN_DB                  0x0008
# define LEAVING_DB  0x0010
# define LEAVING_RH  0x0020
# define LEAVING_LH  0x0040
# define CALLING_RESP           0x0080
# define UNRESERVING            0x0100

# define ALL_STAGES  0x01ff


# define HISTORY_SIZE ((int)( 5000 *   1.2 *     2 *  60 * 2.22222))

# define TRANSACTION_DEBUG_INFO\
  int iStage;\
  DWORD dwThreadId;\
  DWORD dwXPThreadId;\
  int iSynchronous;\
  int iType;\
  int iReserveHistoryId;\
  int iUnreserveHistoryId;\

# define INIT_TRANSACTION(type,pData)\
  gpTransactionPool->iHistoryId++;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iFailure = 0;\
  _ASSERT( gpTransactionPool->iNextFree <= gpTransactionPool->iMaxIndex
);\
  memset( pData, 0x01, gpTransactionPool->iTransactionSize );\
  pData->iStage = 0;\
  pData->dwThreadId = GetCurrentThreadId();\
  pData->dwXPThreadId = 0;\
  pData->iType = type;\
  pData->iReserveHistoryId = gpTransactionPool->iHistoryId;\
  pData->iUnreserveHistoryId = 0;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iOpCode = 1;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iReserveHistoryId = gpTransactionPool->iHistoryId;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iUnreserveHistoryId = 0;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iType = type;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].dwThreadId =
pData->dwThreadId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].dwXPThreadId =
pData->dwXPThreadId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].pTrans = pData;

# define CHECK_TRANSACTION(type,pData)\
  gpTransactionPool->iHistoryId++;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iFailure++;\
  _ASSERT( gpTransactionPool->iNextFree > 0 );\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iFailure++;\
  _ASSERT(((pData->iStage) | ALL_STAGES) == ALL_STAGES);\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iFailure++;\
  if( pData->iSynchronous == 1 )\
    _ASSERT((pData->dwThreadId == GetCurrentThreadId( )));\
  else if( pData->iSynchronous == 0 )\
    _ASSERT((pData->dwXPThreadId == GetCurrentThreadId( )));\
  else\
    _ASSERT(FALSE);\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iFailure++;\
  _ASSERT((pData->iType==type));\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iFailure++;\
  _ASSERT((gpTransactionPool->History[pData->iReserveHistoryId].pTrans)
== pData);\
  pData->iUnreserveHistoryId = gpTransactionPool->iHistoryId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iOpCode = 2;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iReserveHistoryId = pData->iReserveHistoryId;\
  gpTransactionPool->History[gpTransactionPool-
>iHistoryId].iUnreserveHistoryId = gpTransactionPool->iHistoryId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].iType = type;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].dwThreadId =
pData->dwThreadId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].dwXPThreadId =
pData->dwXPThreadId;\
  gpTransactionPool->History[gpTransactionPool->iHistoryId].pTrans = pData;

#else

# define TRANSACTION_DEBUG_INFO

# define INIT_TRANSACTION(type,pData)

# define CHECK_TRANSACTION(type,pData)

#endif

# define NUMBER_POOL_TRANS_TYPES 5
# define DELIVERY_TRANS 0
# define NEW_ORDER_TRANS 1
# define ORDER_STATUS_TRANS 2
# define PAYMENT_TRANS 3
# define STOCK_LEVEL_TRANS 4

#define RESERVE_TRANSACTION_STRUCT(type,pData)\
  EnterCriticalSection( &gpTransactionPool->critSec );\
  pData = gpTransactionPool->index[gpTransactionPool->iNextFree];\
  INIT_TRANSACTION(type,pData);\
  gpTransactionPool->iNextFree++;\
  LeaveCriticalSection( &gpTransactionPool->critSec );

#define UNRESERVE_TRANSACTION_STRUCT(type,pData)\
  EnterCriticalSection( &gpTransactionPool->critSec );\
  CHECK_TRANSACTION(type,pData);\
  gpTransactionPool->index[--gpTransactionPool->iNextFree] = pData;\
  LeaveCriticalSection( &gpTransactionPool->critSec );

typedef struct
{
  CRITICAL_SECTION critSec;
  int           iNextFree;
#ifdef FFE_DEBUG
  int           iMaxIndex;
  int           iTransactionSize;
  int           iHistoryId;
  struct
  {
    int                 iOpCode;
    int                 iFailure;
    int                 iReserveHistoryId;
    int                 iUnreserveHistoryId;
    int                 iType;
    DWORD  dwThreadId;
    DWORD  dwXPThreadId;
    void        *pTrans;
  }                     History[HISTORY_SIZE];
#endif
  void          *index[1];
```

```
  char          data[1];
} TransactionPoolStruct, *pTransactionPoolStruct;

_



typedef void CallersContext;
typedef void *pCallersContext;
typedef void *DBContext;

#define INVALID_DB_CONTEXT NULL

typedef struct _DBDate {
                int year;
                int month;
                int day;
                int hour;
                int minute;
                int second;
} DBDateData, *pDBDateData;



#define CONN_DATA \
   TRANSACTION_DEBUG_INFO\
   int    w_id;\
   int    ld_id;\
   CallersContext *pCC;\
   int    status;\
   int    dbstatus;


typedef struct _ConnData
{
   CONN_DATA
} ConnData, *pConnData;



#define I_DELIVERY \
   CONN_DATA\
   time_t  queue_time;\
   unsigned delta_time;   \
   int    o_carrier_id;

typedef struct _DeliveryDataInput {
   I_DELIVERY
} DeliveryDataInput, *pDeliveryDataInput;

typedef struct _DeliveryData {
   I_DELIVERY
   int    o_id[10];
} DeliveryData, *pDeliveryData;

struct io_order_line {
   int    ol_i_id;
   int    ol_supply_w_id;
   int    ol_quantity;
   char    i_name[25];
   int    s_quantity;
   char    b_g[2];
   double  i_price;
   double  ol_amount;
};

typedef struct _NewOrderData {
   CONN_DATA
   int    d_id;
   int    c_id;
   int    o_ol_cnt;
   int    o_all_local;
   struct io_order_line o_ol[MAX_OL];
   DBDateData o_entry_d;
   char    c_last[17];
   char    c_credit[3];
   double  c_discount;
   double  w_tax;
   double  d_tax;
   int    o_id;
   double  tax_n_discount;
   double  total_amount;
} NewOrderData, *pNewOrderData;

struct status_order_line {
   int    ol_supply_w_id;
   int    ol_i_id;
   int    ol_quantity;
   double  ol_amount;
   DBDateData    ol_delivery_d;
};

typedef struct _OrderStatusData {
   CONN_DATA
   BOOLEAN byname;
   int    d_id;
   int    c_id;
   char    c_last[17];
   char    c_first[17];
   char    c_middle[3];
   double  c_balance;
   int    o_id;
   DBDateData    o_entry_d;
   int    o_carrier_id;
   int    o_ol_cnt;
   struct status_order_line    s_ol[MAX_OL];
} OrderStatusData, *pOrderStatusData;

typedef struct _PaymentData {
   CONN_DATA
   BOOLEAN byname;
   int    d_id;
   int    c_id;
   char    c_last[17];
   int    c_w_id;
   int    c_d_id;
   double  h_amount;
   DBDateData h_date;
   char    w_street_1[21];
   char    w_street_2[21];
   char    w_city[21];
   char    w_state[3];
   char    w_zip[10];
   char    d_street_1[21];
   char    d_street_2[21];
   char    d_city[21];
   char    d_state[3];
   char    d_zip[10];
   char    c_first[17];
   char    c_middle[3];
   char    c_street_1[21];
   char    c_street_2[21];
   char    c_city[21];
   char    c_state[3];
   char    c_zip[10];
   char    c_phone[17];
   DBDateData    c_since;
   char    c_credit[3];
   double  c_credit_lim;
   double  c_discount;
   double  c_balance;
   char    c_data[201];
} PaymentData, *pPaymentData;
```

```
typedef struct _StockLevelData {
   CONN_DATA
   int     threshold;
   int     low_stock;
} StockLevelData, *pStockLevelData;

typedef struct _CheckpointData {
   CONN_DATA
   int     how_many;
   int     interval;
} CheckpointData, *pCheckpointData;



typedef struct _TransactionData {
   int type;
   union {
                DeliveryData delivery;
                NewOrderData newOrder;
                OrderStatusData orderStatus;
                PaymentData payment;
                StockLevelData stockLevel;
       CheckpointData checkpoint;
   }  info;
} TransactionData, *pTransactionData;


typedef struct _TransportData {
  BOOL                  asynchronous;
  BOOL                  generic;
  int                   num_gc;
  int                   num_dy;
  int                   num_no;
  int                   num_os;
  int                   num_pt;
  int                   num_sl;
  BOOL                  dy_use_transport;
  int                   num_dy_servers;
  int                   num_queued_deliveries;
  int                   num_queued_responses;
} TransportData, *pTransportData;


typedef struct _LoginData {
  CONN_DATA
  char                                  szServer[32];
  char                                  szDatabase[32];
  char                                  szUser[32];
  char                                  szPassword[32];
  char                                  szApplication[32];
} LoginData, *pLoginData;

#endif
```

web_ui.c

```
/*+*************************************************************************
 *                                                                         *
 * COPYRIGHT (c) 1997 BY                                                   *
 * COMPAQ COMPUTER CORPORATION, .                                          *
 * ALL RIGHTS RESERVED.                                                    *
 *                                                                         *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE          *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER     *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
 * TRANSFERRED.                                                            *
 •                                                                         *
 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE    *
 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY COMPAQ                   *
 * CORPORATION.                                                            *
 *                                                                         *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                  *
 *                                                                         *
 *                                                                         *
 **************************************************************************/

/*+
 * Abstract: This file contains the Compaq created front end functions
 *                              for the tpcc benchmark.
 *
 * Author: A Bradley & W Carr
 * Creation Date: May 1997
 *
 * Modified history:
 *
 *
*/
#include <windows.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 <crtdbg.h>
#include <process.h>

#define WEB_UI_C
#include <tpccerr.h>
#include <tpccstruct.h>
#include <tpccapi.h>
#include <httpext.h>

#include <tpcc.h>
#include <web_ui.h>
#include <ckpt.h>

#ifdef FFE_DEBUG
# include <crtdbg.h>
static int tmpDbgFlag;
static _HFILE hMemFile;
#endif


#define MAX(a,b) ((a)>(b)?(a):(b))

#define PUT_STRING(szString, iLen, pStart, pStruct) \
pStruct.szStr=szString; pStruct.iIndex=pStart; pStruct.iFieldSize=iLen;

#define CONVERT_SPECIAL(pout,pin,iwid)\
{\
  char *out = pout;\
  char *in = pin;\
  int wid = iwid;\
  while( wid && '\0' != *in )\
  {\
    if( '>' == *in )\
    {*out++='&'; *out++='g'; *out++='t'; *out++=';';}\
    else if( '<' == *in )\
    {*out++='&'; *out++='l'; *out++='t'; *out++=';';}\
    else if( '&' == *in )\
    {*out++='&'; *out++='a'; *out++='m'; *out++='p'; *out++=';';}\
    else if( '\"' == *in )\
    {*out++='&'; *out++='q'; *out++='u'; *out++='o'; *out++='t'; *out++=';';}\
    else\
    {*out++=*in;}\
    in++;\
    wid--;\
  }\
  while( wid-- ) *out++ = ' ';\
}
```

```
#define NO_WDID 0
#define NO_WID NO_WDID + 1
#define NO_DID NO_WID + 1
#define NO_DATE NO_DID + 1
#define NO_CID NO_DATE + 1
#define NO_LAST NO_CID + 1
#define NO_CREDIT NO_LAST + 1
#define NO_DISC NO_CREDIT + 1
#define NO_OID NO_DISC + 1
#define NO_LINES NO_OID + 1
#define NO_W_TAX NO_LINES + 1
#define NO_D_TAX NO_W_TAX + 1
#define NO_S_WID NO_D_TAX + 1
#define NO_IID NO_S_WID + 1
#define NO_INAME NO_IID + 1
#define NO_QTY NO_INAME + 1
#define NO_STOCK NO_QTY + 1
#define NO_BG NO_STOCK + 1
#define NO_PRICE NO_BG + 1
#define NO_AMT NO_PRICE + 1
#define NO_STAT NO_AMT + (14*8) + 1
#define NO_TOTAL NO_STAT + 1


#define PT_WDID_INPUT 0
#define PT_WID_INPUT PT_WDID_INPUT + 1


#define PT_WDID 0
#define PT_LONG_DATE PT_WDID + 1
#define PT_WID PT_LONG_DATE + 1
#define PT_DID PT_WID + 1
#define PT_W_ST_1 PT_DID + 1
#define PT_D_ST_1 PT_W_ST_1 + 1
#define PT_W_ST_2 PT_D_ST_1 + 1
#define PT_D_ST_2 PT_W_ST_2 + 1
#define PT_W_CITY PT_D_ST_2 + 1
#define PT_W_ST PT_W_CITY + 1
#define PT_W_ZIP PT_W_ST + 1
#define PT_D_CITY PT_W_ZIP + 1
#define PT_D_ST PT_D_CITY + 1
#define PT_D_ZIP PT_D_ST + 1
#define PT_CID PT_D_ZIP + 1
#define PT_C_WID PT_CID + 1
#define PT_C_DID PT_C_WID + 1
#define PT_FIRST PT_C_DID + 1
#define PT_MIDDLE PT_FIRST + 1
#define PT_LAST PT_MIDDLE + 1
#define PT_SM_DATE PT_LAST + 1
#define PT_C_STR_1 PT_SM_DATE + 1
#define PT_CREDIT PT_C_STR_1 + 1
#define PT_D_STR_2 PT_CREDIT + 1
#define PT_DISC PT_D_STR_2 + 1
#define PT_C_CITY PT_DISC + 1
#define PT_C_ST PT_C_CITY + 1
#define PT_C_ZIP PT_C_ST + 1
#define PT_C_PHONE PT_C_ZIP + 1
#define PT_AMT PT_C_PHONE + 1
#define PT_BAL PT_AMT + 1
#define PT_LIM PT_BAL + 1
#define PT_CUST_DATA PT_LIM + 1


#define OS_WDID 0
#define OS_WID OS_WDID + 1
#define OS_DID OS_WID + 1
#define OS_CID OS_DID + 1
#define OS_FIRST OS_CID + 1
#define OS_MIDDLE OS_FIRST + 1
#define OS_LAST OS_MIDDLE + 1
#define OS_BAL OS_LAST + 1
#define OS_OID OS_BAL + 1
#define OS_DATE OS_OID + 1
#define OS_CAR_ID OS_DATE + 1
#define OS_S_WID OS_CAR_ID + 1
#define OS_IID OS_S_WID + 1
#define OS_QTY OS_IID + 1
#define OS_AMT OS_QTY + 1
#define OS_SM_DATE OS_AMT + 1

#define D_WDID 0
#define D_WID D_WDID + 1
#define D_CAR D_WID + 1
#define D_QUEUE1 D_CAR + 1
#define D_DELTA1 D_QUEUE1 + 1
#define D_WID1 D_DELTA1 + 1
#define D_CAR1 D_WID1 + 1
#define D_OID10 D_CAR1 + 1
#define D_OID11 D_OID10 + 1
#define D_OID12 D_OID11 + 1
#define D_OID13 D_OID12 + 1
#define D_OID14 D_OID13 + 1
#define D_OID15 D_OID14 + 1
#define D_OID16 D_OID15 + 1
#define D_OID17 D_OID16 + 1
#define D_OID18 D_OID17 + 1
#define D_OID19 D_OID18 + 1
#define D_QUEUE2 D_OID19 + 1
#define D_DELTA2 D_QUEUE2 + 1
#define D_WID2 D_DELTA2 + 1
#define D_CAR2 D_WID2 + 1
#define D_OID20 D_CAR2 + 1
#define D_OID21 D_OID20 + 1
#define D_OID22 D_OID21 + 1
#define D_OID23 D_OID22 + 1
#define D_OID24 D_OID23 + 1
#define D_OID25 D_OID24 + 1
#define D_OID26 D_OID25 + 1
#define D_OID27 D_OID26 + 1
#define D_OID28 D_OID27 + 1
#define D_OID29 D_OID28 + 1


#define SL_WDID 0
#define SL_WID SL_WDID + 1
#define SL_DID SL_WID + 1
#define SL_TH SL_DID + 1
#define SL_LOW SL_TH + 1

#define WDID(w_id,d_id) (w_id*10+(d_id-1))

#define PANIC_FORM_SIZE 4096

#define NUMBER_POOL_FORM_TYPES 5
#define DELIVERY_FORM 0
#define NEW_ORDER_FORM 1
#define ORDER_STATUS_FORM 2
#define PAYMENT_FORM 3
#define STOCK_LEVEL_FORM 4

#define NUMBER_POOL_RESPONSE_TYPES 5
#define DELIVERY_RESPONSE 0
#define NEW_ORDER_RESPONSE 1
#define ORDER_STATUS_RESPONSE 2
#define PAYMENT_RESPONSE 3
#define STOCK_LEVEL_RESPONSE 4

#ifdef FFE_DEBUG
# define FFE_ASSERT(arg) _ASSERT(arg)
#else
# define FFE_ASSERT(arg)
```

```
#endif

#define RESERVE_FORM(type,szForm)\
  {\
    EnterCriticalSection( &gpForms->critSec[type] );\
    FFE_ASSERT( gpForms->iNextFreeForm[type] <= gpForms-
>iMaxIndex[type] );\
    szForm = gpForms->index[gpForms->iFirstFormIndex[type] +\
                                          gpForms->iNextFreeForm[type]++];\
    LeaveCriticalSection( &gpForms->critSec[type] );\
  }
#define UNRESERVE_FORM(type,szForm)\
  {\
    EnterCriticalSection( &gpForms->critSec[type] );\
    FFE_ASSERT( gpForms->iNextFreeForm[type] > 0 );\
    gpForms->index[gpForms->iFirstFormIndex[type] +\
                                --gpForms->iNextFreeForm[type]] = szForm;\
    LeaveCriticalSection( &gpForms->critSec[type] );\
  }

#define RESERVE_RESPONSE(type,szResponse)\
  {\
    EnterCriticalSection( &gpResponses->critSec[type] );\
    FFE_ASSERT(gpResponses->iNextFreeResponse[type]<=gpResponses-
>iMaxIndex[type]);\
    szResponse = gpResponses->index[gpResponses->iFirstResponseIndex[type]
+\
                                          gpResponses-
>iNextFreeResponse[type]++];\
    LeaveCriticalSection( &gpResponses->critSec[type] );\
  }
#define UNRESERVE_RESPONSE(type,szResponse)\
  {\
    EnterCriticalSection( &gpResponses->critSec[type] );\
    FFE_ASSERT(gpResponses->iNextFreeResponse[type] > 0 );\
    gpResponses->index[gpResponses->iFirstResponseIndex[type] +\
                                --gpResponses->iNextFreeResponse[type]] =
szResponse;\
    LeaveCriticalSection( &gpResponses->critSec[type] );\
  }

#define RESERVE_PANIC_FORM(szForm)\
  {\
    EnterCriticalSection( &gpPanicForms->critSec );\
    FFE_ASSERT( gpPanicForms->iNextFree <= gpPanicForms->iMaxIndex );\
    szForm = gpPanicForms->index[gpPanicForms->iNextFree++];\
    LeaveCriticalSection( &gpPanicForms->critSec );\
  }

#define UNRESERVE_PANIC_FORM(szForm)\
  {\
    EnterCriticalSection( &gpPanicForms->critSec );\
    FFE_ASSERT( gpPanicForms->iNextFree > 0 );\
    gpPanicForms->index[--gpPanicForms->iNextFree] = szForm;\
    LeaveCriticalSection( &gpPanicForms->critSec );\
  }

#if 0
CMD                     0
FORM ID                 3
LOGIN WAREHOUSE         4
LOGIN DISTRICT          5
DELI QUEUE TIME         6
CARRIER ID              7
DISTRICT  8
CUSTOMER                9
NEWORDER FIELDS A-X,a-u
CUST LAST NAME          Y
CUST WAREHOUSE          Z
CUST DISTRICT           v
AMOUNT PAID             w
THRESHOLD               x
#endif

#define MENU_BAR \
"<HR>"\
"<INPUT TYPE=submit NAME=0 VALUE=NewOrder>"\
"<INPUT TYPE=submit NAME=0 VALUE=Payment>"\
"<INPUT TYPE=submit NAME=0 VALUE=Delivery>"\
"<INPUT TYPE=submit NAME=0 VALUE=OrderStatus>"\
"<INPUT TYPE=submit NAME=0 VALUE=StockLevel>"\
"<INPUT TYPE=submit NAME=0 VALUE=Exit>"

static char    szFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>";

static char    szWelcomeFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=3 VALUE=W00>"
"Please Identify your Warehouse and District for this session.<BR>"
"Warehouse ID <INPUT NAME=4 SIZE=5><BR>"
"District ID <INPUT NAME=5 SIZE=2><BR>"
"<HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Submit>"
"</FORM></BODY>";

static char
szWelcomeForm[sizeof(szWelcomeFormTemplate)+FILENAMESIZE];
static int  iWelcomeFormLen;

static char    szMainMenuFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=3 VALUE=M%06d>"
"%55.55s<BR>"
"Select Desired Transaction.<BR>"
MENU_BAR
"</FORM></BODY>";

static char szDeliveryFormTemp2i[] =
"<INPUT TYPE=hidden NAME=3 VALUE=D######>"
"<INPUT TYPE=hidden NAME=6 VALUE=0>"
"<PRE>                          Delivery<BR>"
"Warehouse: #####<BR><BR>"
"Carrier Number: <INPUT NAME=7 SIZE=1><BR><BR>"
"Execution Status:<BR></PRE>"
"<HR><INPUT TYPE=submit NAME=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szDeliveryFormTemp2p[] =
"<INPUT TYPE=hidden NAME=3 VALUE=d######>"
"<PRE>                          Delivery<BR>"
"Warehouse: #####<BR><BR>"
"Carrier Number: ##<BR><BR>"
"Execution Status: Delivery has been queued.<BR>"
"Previous Deliveries:<BR>"
"########## ######## ##### ## #### #### #### #### #### #### #### ####
#### ####"
"<BR>"
"########## ######## ##### ## #### #### #### #### #### #### #### ####
#### ####"
"<BR>"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szNewOrderFormTemp2i[] =
"<INPUT TYPE=hidden NAME=3 VALUE=N######>"
"<PRE>                          New Order<BR>"
"Warehouse: #####"
"  District: <INPUT NAME=8 SIZE=2>                    Date:<BR>"
"Customer:  <INPUT NAME=9 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=A SIZE=4>  <INPUT NAME=B SIZE=6>"
"                        <INPUT NAME=C SIZE=1><BR>"
" <INPUT NAME=D SIZE=4>  <INPUT NAME=E SIZE=6>"
"                        <INPUT NAME=F SIZE=1><BR>"
" <INPUT NAME=G SIZE=4>  <INPUT NAME=H SIZE=6>"
"                        <INPUT NAME=I SIZE=1><BR>"
" <INPUT NAME=J SIZE=4>  <INPUT NAME=K SIZE=6>"
"                        <INPUT NAME=L SIZE=1><BR>"
" <INPUT NAME=M SIZE=4>  <INPUT NAME=N SIZE=6>"
"                        <INPUT NAME=O SIZE=1><BR>"
" <INPUT NAME=P SIZE=4>  <INPUT NAME=Q SIZE=6>"
"                        <INPUT NAME=R SIZE=1><BR>"
" <INPUT NAME=S SIZE=4>  <INPUT NAME=T SIZE=6>"
"                        <INPUT NAME=U SIZE=1><BR>"
" <INPUT NAME=V SIZE=4>  <INPUT NAME=W SIZE=6>"
"                        <INPUT NAME=X SIZE=1><BR>"
" <INPUT NAME=a SIZE=4>  <INPUT NAME=b SIZE=6>"
"                        <INPUT NAME=c SIZE=1><BR>"
" <INPUT NAME=d SIZE=4>  <INPUT NAME=e SIZE=6>"
"                        <INPUT NAME=f SIZE=1><BR>"
" <INPUT NAME=g SIZE=4>  <INPUT NAME=h SIZE=6>"
"                        <INPUT NAME=i SIZE=1><BR>"
" <INPUT NAME=j SIZE=4>  <INPUT NAME=k SIZE=6>"
"                        <INPUT NAME=l SIZE=1><BR>"
" <INPUT NAME=m SIZE=4>  <INPUT NAME=n SIZE=6>"
"                        <INPUT NAME=o SIZE=1><BR>"
" <INPUT NAME=p SIZE=4>  <INPUT NAME=q SIZE=6>"
"                        <INPUT NAME=r SIZE=1><BR>"
" <INPUT NAME=s SIZE=4>  <INPUT NAME=t SIZE=6>"
"                        <INPUT NAME=u SIZE=1><BR>"
"Execution Status:                               Total:<BR><HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szNewOrderFormTemp2p[] =
"<INPUT TYPE=hidden NAME=3 VALUE=n######>"
"<PRE>                        New Order<BR>"
"Warehouse: #####"
"  District: ##                    Date: ################### <BR>"
"Customer:  ####   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>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
" ####  ######  ########################  ##   ###   #  "
"$######  $#######  <BR>"
"Execution Status: ########################               "
"Total:  $########  <BR>"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szOrderStatusFormTemp2i[] =
"<INPUT TYPE=hidden NAME=3 VALUE=O######>"
"<PRE>                          Order-Status<BR>"
"Warehouse: #####  District: <INPUT NAME=8 SIZE=1><BR>"
"Customer: <INPUT NAME=9 SIZE=4>   Name:                "
"<INPUT NAME=Y 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=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szOrderStatusFormTemp2p[] =
"<INPUT TYPE=hidden NAME=3 VALUE=o######>"
"<PRE>                          Order-Status<BR>"
"Warehouse: #####  District: ##<BR>"
"Customer: ####   Name: ################ ## ################<BR>"
"Cust-Balance: $#########<BR><BR>"
"Order-Number: ########   Entry-Date: ###################   Carrier-
Number: ##"
"<BR>"
"Supply-W     Item-Id    Qty     Amount      Delivery-Date<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"  ####      ######     ##     $########     ##########<BR>"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szPaymentFormTemp2i[] =
"<INPUT TYPE=hidden NAME=3 VALUE=P######>"
"<PRE>                          Payment<BR>"
"Date:  <BR><BR>"
"Warehouse: #####                    District: <INPUT NAME=8 SIZE=2><BR>"
"<BR><BR><BR><BR>"
"Customer: <INPUT NAME=9 SIZE=4>"
"Cust-Warehouse: <INPUT NAME=Z SIZE=5>  "
"Cust-District: <INPUT NAME=v SIZE=1><BR>"
"Name:                 <INPUT NAME=Y SIZE=16>                "
"Since:<BR>"
"                                   Credit:<BR>"
"                                   Disc:<BR>"
"                                   Phone:<BR><BR>"
"Amount Paid:        $<INPUT NAME=w SIZE=7>      New Cust Balance:<BR>"
"Credit Limit:<BR><BR>Cust-Data: <BR><BR><BR><BR></PRE><HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Process>"
```

```
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szPaymentFormTemp2p[] =
"<INPUT TYPE=hidden NAME=3 VALUE=p######>"
"<PRE>                          Payment<BR>"
"Date: ################## <BR><BR>"
"Warehouse: #####                  District: ##<BR>"
"####################                ####################<BR>"
"####################                ####################<BR>"
"################### ## ##########      ################### ##
##########"
"<BR><BR>"
"Customer: ####  Cust-Warehouse: ##### Cust-District: ##<BR>"
"Name:   ################ ## ################     Since:
##########<BR>"
"       ####################                Credit: ##<BR>"
"       ####################                %Disc:  #####<BR>"
"       ################### ## ##########       Phone:
##################"
"<BR><BR>"
"Amount Paid:         $#######      New Cust Balance: $##############<BR>"
"Credit Limit:   $##############<BR><BR>"
"Cust-Data:
##################################################<BR>"
"         ##################################################<BR>"
"         ##################################################<BR>"
"         ##################################################<BR>"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szStockLevelFormTemp2i[] =
"<INPUT TYPE=hidden NAME=3 VALUE=S######>"
"<PRE>                          Stock-Level<BR>"
"Warehouse: #####  District: ##<BR><BR>"
"Stock Level Threshold: <INPUT NAME=x SIZE=2><BR><BR>"
"low stock:    <BR><HR>"
"<INPUT TYPE=submit NAME=0 VALUE=Process>"
"<INPUT TYPE=submit NAME=0 VALUE=Menu>"
"</FORM></BODY>";

static char szStockLevelFormTemp2p[] =
"<INPUT TYPE=hidden NAME=3 VALUE=s######>"
"<PRE>                          Stock-Level<BR>"
"Warehouse: #####  District: ##<BR><BR>"
"Stock Level Threshold: ##<BR><BR>"
"low stock: ###"
"</PRE>"
MENU_BAR
"</FORM></BODY>";

static char szErrorFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden NAME=3 VALUE=e%06d>"
"Error: %d (%s): %s"
MENU_BAR
"</FORM></BODY>";

static char szResponseHeaderTemplate[] =
"Connection: keep-alive\r\n"
"Content-type: text/html\r\n"
"Content-length: ####\r\n"
"\r\n";
static char  szResponseHeader[sizeof(szResponseHeaderTemplate)];
FORM_INDEXES responseHeaderIndexes[1] = { 0 };
int                 responseHeaderLen = 0;


#define MATCHES_BEGIN(p) ('B'==p[0])
#define MATCHES_CHECKPOINT(p) \
(0==strncmp(p,"Checkpoint",strlen("Checkpoint")))
#define MATCHES_CHECKPOINT_STARTUP(p) \
(0==strncmp(p,"CheckpointStartup",strlen("CheckpointStartup")))
#define MATCHES_CLEAR(p) ('C'==p[0]&&'l'==p[1])
#define MATCHES_DELIVERY(p) ('D'==p[0])
#define MATCHES_EXIT(p) ('E'==p[0])
#define MATCHES_MENU(p) ('M'==p[0])
#define MATCHES_NEWORDER(p) ('N'==p[0])
#define MATCHES_ORDERSTATUS(p) ('O'==p[0])
#define MATCHES_PAYMENT(p) ('P'==p[0]&&'a'==p[1])
#define MATCHES_PROCESS(p) ('P'==p[0]&&'r'==p[1])
#define MATCHES_STOCKLEVEL(p) ('S'==p[0]&&'t'==p[1])
#define MATCHES_SUBMIT(p) ('S'==p[0]&&'u'==p[1])
#ifdef FFE_DEBUG
# define MATCHES_MEMORYCHECK(p) ('!'==p[0]&&'M'==p[1])
#endif


void BeginCmd( EXTENSION_CONTROL_BLOCK *pECB );
void CheckpointCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id );
void CheckpointStartupCmd( EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id );
void ClearCmd( EXTENSION_CONTROL_BLOCK *pECB );
void ExitCmd( EXTENSION_CONTROL_BLOCK *pECB );
void MenuCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id
);
void SubmitCmd( EXTENSION_CONTROL_BLOCK *pECB, int *w_id, int
*ld_id );
void MemoryCheckCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id );

BOOL GetKeyValuePtr( char *szIPtr, char *szKey, char **pszOPtr );
BOOL GetCharKeyValuePtr( char *szIPtr, char cKey, char **pszOPtr );
BOOL GetKeyValueString( char *szIPtr, char *szKey,
                                        char *szValue, int iSize );
BOOL GetWDID(char *ptr, int *lw_id, int *ld_id, char **optr);

void Log( char *szType, char *szStr );
void MakePanicPool( DWORD dwResponseSize );
void MakeTemplatePool( DWORD dwFormSize, DWORD dwResponseSize );
void MakeTransactionPool( DWORD dwTransactionPoolSize );
void DeletePanicPool( void );
void DeleteTemplatePool( void );
void DeleteTransactionPool( void );

int ProcessDeliveryQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                        char *lpszQueryString,
                                        int w_id, int ld_id );
int ProcessNewOrderQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                        char *lpszQueryString,
                                        int w_id, int ld_id );
int ProcessOrderStatusQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                          char *lpszQueryString,
                                          int w_id, int ld_id );
int ProcessPaymentQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                        char *lpszQueryString,
                                        int w_id, int ld_id );
int ProcessStockLevelQuery( EXTENSION_CONTROL_BLOCK *pECB,
                                          char *lpszQueryString,
                                          int w_id, int ld_id );

DWORD ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB);

void PutNumeric( int iInt, int iFieldSize, char *pChar );
void SendErrorResponse( EXTENSION_CONTROL_BLOCK *pECB, int
iError,
                                        int iErrorType, char *szMsg, int w_id, int
ld_id,
                                        pConnData pConn );
void SendMainMenuForm( EXTENSION_CONTROL_BLOCK *pECB,
                                    int w_id, int ld_id, char *szStatus );
```

```
void SendResponse(EXTENSION_CONTROL_BLOCK *pECB, char *szStr, int
iStrLen);
void SendWelcomeForm(EXTENSION_CONTROL_BLOCK *pECB);
#ifdef FFE_DEBUG
unsigned __stdcall CheckMemory(void *param);
#endif


typedef struct
{
  char *szStr;
  int iIndex;
  int iFieldSize;
  int iNewIndex;
  int iNewFieldSize;
} PutStrStruct, *pPutStrStruct;

typedef struct
{
  CRITICAL_SECTION critSec;
#ifdef FFE_DEBUG
  int           iMaxIndex;
#endif
  int           iNextFree;
  char          *index[1];
  char          forms[PANIC_FORM_SIZE];
} PanicStruct, *pPanicStruct;

typedef struct
{
  CRITICAL_SECTION critSec[NUMBER_POOL_FORM_TYPES];
#ifdef FFE_DEBUG
  int           iMaxIndex[NUMBER_POOL_FORM_TYPES];
#endif
  int           iNextFreeForm[NUMBER_POOL_FORM_TYPES];
  int           iFirstFormIndex[NUMBER_POOL_FORM_TYPES];
  char          *index[1];
  char          forms[1];
} FormStruct, *pFormStruct;

typedef struct
{
  CRITICAL_SECTION critSec[NUMBER_POOL_RESPONSE_TYPES];
#ifdef FFE_DEBUG
  int           iMaxIndex[NUMBER_POOL_RESPONSE_TYPES];
#endif
  int           iNextFreeResponse[NUMBER_POOL_RESPONSE_TYPES];
  int           iFirstResponseIndex[NUMBER_POOL_RESPONSE_TYPES];
  char          *index[1];
  char          responses[1];
} ResponseStruct, *pResponseStruct;


static int                  iInitStatus    = ERR_SUCCESS;

static CRITICAL_SECTION startupCriticalSection;
static BOOL           startupFlag    = FALSE;

static pPanicStruct gpPanicForms       = NULL;
static int giPanic                                  = 0;
static pFormStruct gpForms                          = 0;
static int giFormLen[NUMBER_POOL_FORM_TYPES] = { 0 };
static pResponseStruct gpResponses    = 0;
static int giResponseLen[NUMBER_POOL_RESPONSE_TYPES] = { 0 };



BOOL APIENTRY
DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpRes)
{
  char                  szTmpFileName[FILENAMESIZE];
  DWORD                 dwFileNameLen;
  char                  *pChr;

  switch( ul_reason_for_call )
  {
  case DLL_PROCESS_ATTACH:
#ifdef FFE_DEBUG
    tmpDbgFlag = _CrtSetDbgFlag(_CRTDBG_REPORT_FLAG);
    tmpDbgFlag |= _CRTDBG_CHECK_CRT_DF;
    tmpDbgFlag |= _CRTDBG_LEAK_CHECK_DF;
    tmpDbgFlag |= _CRTDBG_ALLOC_MEM_DF;
    tmpDbgFlag |= _CRTDBG_CHECK_ALWAYS_DF;
    _CrtSetDbgFlag(tmpDbgFlag);

    hMemFile = CreateFile( "MemErrors", GENERIC_WRITE,
FILE_SHARE_READ, NULL,
                                        OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL );

    _CrtSetReportMode( _CRT_WARN, _CRTDBG_MODE_FILE );
    _CrtSetReportFile( _CRT_WARN, hMemFile );
    _CrtSetReportMode( _CRT_ERROR, _CRTDBG_MODE_FILE );
    _CrtSetReportFile( _CRT_ERROR, hMemFile );
    _CrtSetReportMode( _CRT_ASSERT, _CRTDBG_MODE_FILE );
    _CrtSetReportFile( _CRT_ASSERT, hMemFile );
#endif
#ifdef DEBUG_ENTRY
    DebugBreak( );
#endif
    InitializeCriticalSection( &startupCriticalSection );

    dwFileNameLen = GetModuleFileName( hModule, szTmpFileName,
FILENAMESIZE-1);
    if( 0 == dwFileNameLen )
      return FALSE;

    pChr = strrchr( szTmpFileName, '\\' );
    if( NULL == pChr )
      return FALSE;

    pChr++;
    dwFileNameLen = strlen( pChr );
    if( 0 >= dwFileNameLen )
      return FALSE;

    CopyMemory( szModName, pChr, dwFileNameLen+1 );

    iWelcomeFormLen = sprintf(szWelcomeForm, szWelcomeFormTemplate,
szModName);

    if( ERR_SUCCESS != iInitStatus )
      return TRUE;

    break;
  case DLL_THREAD_ATTACH:
    break;
  case DLL_THREAD_DETACH:
    dwFileNameLen = 0;
    break;
  case DLL_PROCESS_DETACH:

    TPCCShutdown( );

    DeleteTransactionPool( );
    DeleteTemplatePool( );
    DeletePanicPool( );

    DeleteCriticalSection( &startupCriticalSection );

    TPCCCloseLog( );

    break;
  }
```

```
 return TRUE;
}



BOOL WINAPI
GetExtensionVersion(HSE_VERSION_INFO *pVersion)
{

 pVersion->dwExtensionVersion =
  MAKELONG(HSE_VERSION_MINOR, HSE_VERSION_MAJOR);
 strncpy(pVersion->lpszExtensionDesc,
                "Compaq TPC-C Server.",
                HSE_MAX_EXT_DLL_NAME_LEN-1);
 *(pVersion->lpszExtensionDesc) = '\0';

 return TRUE;
}



DWORD WINAPI
HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
 DWORD                          status;
 int                            dbstatus;
 TransportData                  transport;
 int                            formPoolSize;
 int                            responsePoolSize;
 int                            panicPoolSize;
 int                            transactionPoolSize;
 int                            transportThreadLimit;
 int                            webServerThreadLimit;
#ifdef FFE_DEBUG
 unsigned long                  ulHandle;
 unsigned               uThreadAddr;
#endif

 if ( ! startupFlag ) {
  EnterCriticalSection( &startupCriticalSection );
  if ( ! startupFlag ) {
   if( ERR_SUCCESS == iInitStatus ) {
            if ( ERR_SUCCESS != ( iInitStatus = ReadRegistrySettings( )))
             MakePanicPool( 50 );
            else {
             TPCCOpenLog( );
             iInitStatus = TPCCGetTransportData( &transport );
             if( ERR_SUCCESS != iInitStatus ) {
              MakePanicPool( 50 );
             }
             else {
              webServerThreadLimit = gdwPoolThreadLimit;
              if( transport.generic ) {
               transportThreadLimit = transport.num_gc;
              }
              else {
               transportThreadLimit = transport.num_dy +
                        transport.num_no + transport.num_os +
                        transport.num_pt + transport.num_sl;
              }



              transactionPoolSize = transport.num_dy_servers +
               transport.num_queued_deliveries +
transport.num_queued_responses;



              formPoolSize = webServerThreadLimit;
              if( transport.asynchronous ) {



               responsePoolSize =
MAX(webServerThreadLimit,transportThreadLimit);



               panicPoolSize = MAX( webServerThreadLimit,
transportThreadLimit );


               transactionPoolSize +=
                        webServerThreadLimit + transportThreadLimit;
              }
              else {


               responsePoolSize = webServerThreadLimit;
               panicPoolSize = webServerThreadLimit;



               transactionPoolSize +=
                        MAX( webServerThreadLimit, transportThreadLimit );
              }

              MakeTemplatePool( formPoolSize, responsePoolSize );
              MakePanicPool( panicPoolSize );
              MakeTransactionPool( transactionPoolSize );
              dbstatus = TPCCStartup( iMaxConnections, &transport );
              if( ERR_DB_SUCCESS != dbstatus ) {
               iInitStatus = dbstatus;
              }

#ifdef FFE_DEBUG
                        ulHandle = _beginthreadex(
                                 NULL,
                                 0,

                                 CheckMemory,
                                 NULL,
                                 0,

                                 &uThreadAddr);

                        _ASSERT (ulHandle != 0);
#endif
             }
            }
   }
   startupFlag = TRUE;
  }
  LeaveCriticalSection( &startupCriticalSection );
 }

 if( ERR_SUCCESS != iInitStatus )
 {
  SendErrorResponse(pECB, iInitStatus, ERR_TYPE_WEBDLL, NULL, -1, -1,
NULL);
  return HSE_STATUS_SUCCESS;
 }
```

```
 if ( bLog )
 {
  TPCCLog( "%s %s\r\n", "* QUERY    *", pECB->lpszQueryString );
 }


 status = ProcessQueryString(pECB);


 return status;
}



void
SendErrorResponse( EXTENSION_CONTROL_BLOCK *pECB, int iError, int
iErrorType,
                                char *szMsg, int w_id, int ld_id, pConnData pConn )
{
 int             ii;

 static char szNoMsg[] = "";
 char                        *szErrorTypeMsg;
 char                        *szErrorMsg;
 char                        *szForm;
 int                         iStrLen;

 if ( !szMsg  )
  szMsg = szNoMsg;

 RESERVE_PANIC_FORM( szForm );

 if( ERR_TYPE_WEBDLL == iErrorType )
 {
  ii = 0;
  while( '\0' != errorMsgs[ii].szMsg[0] && iError != errorMsgs[ii].iError )
    ii++;
  if ( '\0' == errorMsgs[ii].szMsg[0] )
    ii = 1;
  szErrorTypeMsg = "TPCCWEB";
  szErrorMsg = errorMsgs[ii].szMsg;
 }
 else if( ERR_TYPE_SQL == iErrorType )
 {
  szErrorTypeMsg = "SQLSVR";
  szErrorMsg = szMsg;
 }
 else if( ERR_TYPE_DBLIB == iErrorType )
 {
  szErrorTypeMsg = "DBLIB";
  szErrorMsg = szMsg;
 }

 if( NULL != pConn )
  TPCCTransactionErr( pConn, "%s(%d): %s\r\n",
                                        szErrorTypeMsg, iError, szErrorMsg );
 else
  TPCCErr( "%s(%d): %s\r\n", szErrorTypeMsg, iError, szErrorMsg );

 iStrLen = sprintf( szForm, szErrorFormTemplate, szModName,
                                WDID(w_id,ld_id), iError, szErrorTypeMsg,
szErrorMsg );

 SendResponse(pECB, szForm, iStrLen);

 UNRESERVE_PANIC_FORM( szForm );
}



void
HandlePanic( pPutStrStruct pStruct,
                    char *szInput, int iInputSize,
                    char **szOutput, int *iOutputSize )
{
 pPutStrStruct pStructTmp1;
 pPutStrStruct pStructTmp2;
 char *pIChar;
 int iExtra;
 int iTotalExtra;
 char *szTmp;

 RESERVE_PANIC_FORM( szTmp );


 *szOutput = szTmp;
 memcpy( szTmp, szInput, pStruct->iIndex );


 pStructTmp1 = pStruct;
 while( NULL != pStructTmp1->szStr ) {
  pStructTmp1->iNewIndex = pStructTmp1->iIndex;
  pStructTmp1->iNewFieldSize = pStructTmp1->iFieldSize;
  pStructTmp1++;
 }


 pStructTmp1 = pStruct;
 iTotalExtra = 0;
 while( NULL != pStructTmp1->szStr ) {
  pIChar = pStructTmp1->szStr;
  iExtra = 0;
  while( 0 != *pIChar )
  {
   if( '"' == *pIChar )
                iExtra += 5;
   else if( '&' == *pIChar )
                iExtra += 4;
   else if( '<' == *pIChar )
                iExtra += 3;
   else if( '>' == *pIChar )
                iExtra += 3;
   pIChar++;
  }


  pStructTmp1->iNewFieldSize += iExtra;


  for( pStructTmp2 = pStructTmp1+1;
               NULL != pStructTmp2->szStr;
               pStructTmp2++ )
   pStructTmp2->iNewIndex += iExtra;

  pStructTmp1++;
  iTotalExtra += iExtra;
 }


 *iOutputSize = iInputSize + iTotalExtra;


 --pStructTmp1;
 memcpy( &szTmp[pStructTmp1->iNewIndex + pStructTmp1-
>iNewFieldSize],
               &szInput[pStructTmp1->iIndex + pStructTmp1->iFieldSize],
               iInputSize - pStructTmp1->iIndex + pStructTmp1->iFieldSize);


 pStructTmp2 = pStructTmp1--;
 while( pStruct != pStructTmp2 )
 {
```

```
   memcpy( &szTmp[pStructTmp1->iNewIndex + pStructTmp1->iNewFieldSize],
                  &szInput[pStructTmp1->iIndex + pStructTmp1->iFieldSize],
                  pStructTmp2->iIndex -
                  ( pStructTmp1->iIndex + pStructTmp1->iFieldSize ));
   pStructTmp2 = pStructTmp1--;
  }


  pStructTmp1 = pStruct;
  while( NULL != pStructTmp1->szStr ) {
   CONVERT_SPECIAL( &szTmp[pStructTmp1->iNewIndex], pStructTmp1->szStr,
                                 pStructTmp1->iNewFieldSize );
   pStructTmp1++;
  }
}



void
SendResponse(EXTENSION_CONTROL_BLOCK *pECB, char *szForm, int iStrLen)
{
  char                  szHeader1[sizeof(szResponseHeader)];
  static char  szHeader[] = "200 Ok";
  static int    iSize = sizeof(szHeader)-1;
  int                   lpbSize;

  lpbSize = iStrLen + 1;

  if ( bLog )
   TPCCLog( "%s %s\r\n", "* RESPONSE *", szForm );

  CopyMemory( szHeader1, szResponseHeader, sizeof(szResponseHeader) );
  PutNumeric(lpbSize, responseHeaderIndexes[0].iLen,
                  &szHeader1[responseHeaderIndexes[0].iStartIndex]);

  (*pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_SEND_RESPONSE_HEADER,
                                                szHeader, &iSize,
(LPDWORD)szHeader1);
  (*pECB->WriteClient)(pECB->ConnID, szForm, &lpbSize, 0);
}



void
ParseTemplateString(char *szForm, int *pcurLen,
                  char *formTemplate, FORM_INDEXES *indexes)
{
  int   curIndex = 0;
  int           ii = 0;
  int           jj;
  int           curLen;

  curLen = *pcurLen;
  while ('\0' != formTemplate[ii])
  {
   if('#' != formTemplate[ii])
   {
    szForm[curLen] = formTemplate[ii];
    ii++;
    curLen++;
   }
   else
   {
    jj = 0;
    indexes[curIndex].iStartIndex = curLen;
    while('#' == formTemplate[ii])
    {
                  jj++;
                  szForm[curLen] = formTemplate[ii];
                  curLen++;
                  ii++;
    }
    indexes[curIndex].iLen = jj;
    curIndex++;
   }
  }
  szForm[curLen] = '\0';
  *pcurLen = curLen;
}



void
PutNumeric( int iInt, int iFieldSize, char *pChar )
{
  int iSaveSize = iFieldSize;
  char *pSaveStart = pChar;
  char pAsterisk[] = "**********";
  BOOL bSignFlag = TRUE;

  pChar += (iFieldSize - 1);
  if(0 > iInt)
  {
   bSignFlag = FALSE;
   iInt = abs(iInt);
  }

  do
  {
   *pChar = ( iInt % 10 ) + '0';
   iInt /= 10;
   iFieldSize--;
   if( iFieldSize )
    pChar--;
  } while( iFieldSize );

  if( !bSignFlag )
  {
   if('0' == *pChar)
    *pChar = '-';
   else
   {
    memcpy( pSaveStart, pAsterisk, iSaveSize );
    return;
   }
  }

  if( 0 != iInt )
  {

   memcpy( pSaveStart, pAsterisk, iSaveSize );
  }
}



void
SendDeliveryForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id )
{
  char          *deliveryForm;

  RESERVE_FORM( DELIVERY_FORM, deliveryForm );

  PutNumeric(WDID(w_id,ld_id),
                  deliveryFormIndexesI[D_WDID].iLen,
                  &deliveryForm[deliveryFormIndexesI[D_WDID].iStartIndex]);
  PutNumeric(w_id,
                  deliveryFormIndexesI[D_WID].iLen,
                  &deliveryForm[deliveryFormIndexesI[D_WID].iStartIndex]);
```

```
 SendResponse(pECB, deliveryForm, giFormLen[DELIVERY_FORM]);

 UNRESERVE_FORM( DELIVERY_FORM, deliveryForm );
}



void
SendNewOrderForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
 char          *newOrderForm;

 RESERVE_FORM( NEW_ORDER_FORM, newOrderForm );

 PutNumeric(WDID(w_id,ld_id),
                  newOrderFormIndexes[NO_WDID].iLen,

&newOrderForm[newOrderFormIndexes[NO_WDID].iStartIndex]);
 PutNumeric(w_id,
                  newOrderFormIndexes[NO_WID].iLen,

&newOrderForm[newOrderFormIndexes[NO_WID].iStartIndex]);

 SendResponse(pECB, newOrderForm, giFormLen[NEW_ORDER_FORM]);

 UNRESERVE_FORM( NEW_ORDER_FORM, newOrderForm );
}



void
SendPaymentForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
 char          *paymentForm;

 RESERVE_FORM( PAYMENT_FORM, paymentForm );

 PutNumeric(WDID(w_id,ld_id),
                  paymentFormIndexes[PT_WDID_INPUT].iLen,

&paymentForm[paymentFormIndexes[PT_WDID_INPUT].iStartIndex]);

 PutNumeric(w_id,
                  paymentFormIndexes[PT_WID_INPUT].iLen,

&paymentForm[paymentFormIndexes[PT_WID_INPUT].iStartIndex]);

 SendResponse(pECB, paymentForm, giFormLen[PAYMENT_FORM]);

 UNRESERVE_FORM( PAYMENT_FORM, paymentForm );
}



void
SendOrderStatusForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
 char          *orderStatusForm;

 RESERVE_FORM( ORDER_STATUS_FORM, orderStatusForm );

 PutNumeric(WDID(w_id,ld_id),
                  orderStatusFormIndexes[OS_WDID].iLen,

&orderStatusForm[orderStatusFormIndexes[OS_WDID].iStartIndex]);
 PutNumeric(w_id,
                  orderStatusFormIndexes[OS_WID].iLen,

&orderStatusForm[orderStatusFormIndexes[OS_WID].iStartIndex]);
 SendResponse(pECB, orderStatusForm,
giFormLen[ORDER_STATUS_FORM]);

 UNRESERVE_FORM( ORDER_STATUS_FORM, orderStatusForm );
}



void
SendStockLevelForm( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
d_id )
{
 char          *stockLevelForm;

 RESERVE_FORM( STOCK_LEVEL_FORM, stockLevelForm );

 PutNumeric(WDID(w_id,d_id),
                  stockLevelFormIndexes[SL_WDID].iLen,

&stockLevelForm[stockLevelFormIndexes[SL_WDID].iStartIndex]);
 PutNumeric(w_id,
                  stockLevelFormIndexes[SL_WID].iLen,

&stockLevelForm[stockLevelFormIndexes[SL_WID].iStartIndex]);
 PutNumeric(d_id,
                  stockLevelFormIndexes[SL_DID].iLen,

&stockLevelForm[stockLevelFormIndexes[SL_DID].iStartIndex]);

 SendResponse(pECB, stockLevelForm, giFormLen[STOCK_LEVEL_FORM]);

 UNRESERVE_FORM( STOCK_LEVEL_FORM, stockLevelForm );
}



void
SendMainMenuForm( EXTENSION_CONTROL_BLOCK *pECB,
                        int w_id, int ld_id, char *szStatus )
{
 char          *szForm;
 int           iStrLen;
 static char   *szNoStatus = "";
 char          *pszStatus;

 pszStatus = ( NULL == szStatus ) ? szNoStatus : szStatus;

 RESERVE_PANIC_FORM( szForm );

 iStrLen = sprintf( szForm, szMainMenuFormTemplate,
                        szModName, WDID(w_id,ld_id), pszStatus );

 SendResponse(pECB, szForm, iStrLen);

 UNRESERVE_PANIC_FORM( szForm );
}



void
SendWelcomeForm(EXTENSION_CONTROL_BLOCK *pECB)
{
 SendResponse( pECB, szWelcomeForm, iWelcomeFormLen );
}



DWORD
ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB)
```

```
{
 static char *beginptr = "Begin";
 char *ptr;
 char *cmdptr;
 int cFormID;
 int w_id;
 int ld_id;
 DWORD status;
 int retcode;

 w_id = 0;
 ld_id = 0;

 if ( GetCharKeyValuePtr( pECB->lpszQueryString, '3', &ptr ))
 {
  cFormID = *ptr++;
  if( !GetWDID( ptr, &w_id, &ld_id, &ptr )) {
   SendErrorResponse( pECB, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL,
NULL,
                                        w_id, ld_id, NULL );
   return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }
 }
 else
  cFormID = '\0';


 if ( !GetCharKeyValuePtr( ptr, '0', &cmdptr ))
 {
  if( 0 == strlen( pECB->lpszQueryString )) {
   cmdptr = beginptr;
  }
  else {
   SendErrorResponse( pECB, ERR_COMMAND_UNDEFINED,
ERR_TYPE_WEBDLL,
                                        NULL, w_id, ld_id, NULL );
   return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }
 }

 if( '\0' == cFormID && !MATCHES_BEGIN( cmdptr )) {
  SendErrorResponse( pECB,
ERR_INVALID_FORM_AND_CMD_NOT_BEGIN,
                                ERR_TYPE_WEBDLL, NULL, w_id, ld_id,
NULL );
  return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
 }

 status = HSE_STATUS_SUCCESS_AND_KEEP_CONN;
 if( MATCHES_PROCESS( cmdptr ))
 {
  if( 'N' == cFormID )
   retcode = ProcessNewOrderQuery( pECB, ptr, w_id, ld_id );
  else if( 'P' == cFormID )
   retcode = ProcessPaymentQuery( pECB, ptr, w_id, ld_id );
  else if( 'D' == cFormID )
   retcode = ProcessDeliveryQuery( pECB, ptr, w_id, ld_id );
  else if( 'O' == cFormID )
   retcode = ProcessOrderStatusQuery( pECB, ptr, w_id, ld_id );
  else if( 'S' == cFormID )
   retcode = ProcessStockLevelQuery( pECB, ptr, w_id, ld_id );
  else {
   SendErrorResponse( pECB, ERR_INVALID_FORM,
ERR_TYPE_WEBDLL, NULL,
                                        w_id, ld_id, NULL );
   return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }

  if( ERR_DB_PENDING == retcode )
   status = HSE_STATUS_PENDING;
  else if( ERR_DB_SUCCESS != retcode ) {
   SendErrorResponse( pECB, retcode, ERR_TYPE_WEBDLL,
                                        NULL, w_id, ld_id, NULL );
   return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }
 }
 else if( MATCHES_PAYMENT( cmdptr ))
  SendPaymentForm( pECB, w_id, ld_id );
 else if( MATCHES_NEWORDER( cmdptr ))
  SendNewOrderForm( pECB, w_id, ld_id );
 else if( MATCHES_DELIVERY( cmdptr ))
  SendDeliveryForm( pECB, w_id, ld_id );
 else if( MATCHES_ORDERSTATUS( cmdptr ))
  SendOrderStatusForm( pECB, w_id, ld_id );
 else if( MATCHES_STOCKLEVEL( cmdptr ))
  SendStockLevelForm( pECB, w_id, ld_id );
 else if( MATCHES_EXIT( cmdptr ))
  ExitCmd( pECB );
 else if( MATCHES_SUBMIT( cmdptr ))
  SubmitCmd( pECB, &w_id, &ld_id );
 else if( MATCHES_BEGIN( cmdptr ))
  BeginCmd( pECB );
 else if( MATCHES_MENU( cmdptr ))
  MenuCmd( pECB, w_id, ld_id );
 else if( MATCHES_CLEAR( cmdptr ))
  ClearCmd( pECB );
 else if( MATCHES_CHECKPOINT_STARTUP( cmdptr ))
  CheckpointStartupCmd( pECB, w_id, ld_id );
 else if( MATCHES_CHECKPOINT( cmdptr ))
  CheckpointCmd( pECB, w_id, ld_id );
#ifdef FFE_DEBUG
 else if( MATCHES_MEMORYCHECK( cmdptr ))
  MemoryCheckCmd( pECB, w_id, ld_id );
#endif
 else
  SendErrorResponse( pECB, ERR_COMMAND_UNDEFINED,
ERR_TYPE_WEBDLL,
                                NULL, w_id, ld_id, NULL );

 return status;
}



void
PutFloat2( double dVal, int iFieldSize, char *pChar )
{
 int iInt;
 int iDecimal;
 BOOL bSignFlag = TRUE;
 int iSaveSize = iFieldSize;
 char *pSaveStart = pChar;
 char pAsterisk[] = "**********";

 pChar += (iFieldSize - 1);

 if(0 > dVal)
 {
  bSignFlag = FALSE;
  iInt = abs((int)( dVal * 100. ));
 }
 else
 {
  iInt = (int)( dVal * 100. );
 }
 iDecimal = 2;
 do
 {
  *pChar-- = ( iInt % 10 ) + '0';
  iInt /= 10;
  iFieldSize--;
 } while( --iDecimal );

 *pChar-- = '.';
```

```
  iFieldSize--;

  do
  {
   *pChar-- = ( iInt % 10 ) + '0';
   iInt /= 10;
   iFieldSize--;
  } while( iFieldSize && iInt != 0 );

  if( !iFieldSize && iInt != 0 )
  {

   memcpy(pSaveStart, pAsterisk, iSaveSize);
   return;
  }
  if(!bSignFlag)
  {
   iFieldSize--;
   if( 0 >= iFieldSize )
   {

    memcpy(pSaveStart, pAsterisk, iSaveSize);
    return;
   }
   *pChar-- = '-';
  }


  while( iFieldSize-- )
   *pChar-- = ' ';
}


void
PutHTMLStrings( pPutStrStruct pStruct,
                char *szInput, int iInputSize,
                char **szOutput, int *iOutputSize )
{
  char *pIChar;
  char *pOChar;
  int           iFieldSize;

  while( NULL != pStruct->szStr )
  {
   pIChar = pStruct->szStr;
   pOChar = szInput + pStruct->iIndex;
   iFieldSize = pStruct->iFieldSize;
   while( 0 != *pIChar && iFieldSize )
   {


    if( '>' > *pIChar )
    {
          if( '"' == *pIChar || '&' == *pIChar ||
              '<' == *pIChar || '>' == *pIChar )
          {



           InterlockedIncrement( &giPanic );
           HandlePanic( pStruct, szInput, iInputSize, szOutput, iOutputSize );
           return;
          }
          else
           *pOChar = *pIChar;
    }
    else
          *pOChar = *pIChar;

    pIChar++;
    pOChar++;
    iFieldSize--;
   }


   while( iFieldSize-- )
    *pOChar++ = ' ';

   pStruct++;
  }


  *szOutput = szInput;
  *iOutputSize = iInputSize;

  return;
}



void
TPCCDeliveryResponse( int retcode, pDeliveryData pDelivery,
                      pDeliveryData
CompletedDeliveries[DELIVERY_RESPONSE_COUNT] )
{
  int ssCnt = 0;
  char *szOutput;
  int iOutputLen;
  PutStrStruct StrStruct[2];
  char *deliveryForm;
  int ii, jj, index;
  pDeliveryData pCompletedDelivery;
  static DeliveryData blankDelivery = { 0 };
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = pDelivery->pCC;

  if ( ERR_DB_PENDING == retcode )
  {
   return;
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
    for( jj = 0; jj < DELIVERY_RESPONSE_COUNT; jj++ )
     if( NULL != CompletedDeliveries[jj] ) {
#ifdef FFE_DEBUG
          CompletedDeliveries[jj]->iStage |= UNRESERVING;
#endif
          UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS,
CompletedDeliveries[jj] );
     }

    SendErrorResponse( pECB, ERR_DELIVERY_NOT_PROCESSED,
                             ERR_TYPE_WEBDLL, NULL,
                             pDelivery->w_id, pDelivery->ld_id,
                             (pConnData)pDelivery );

   return;
  }
  else          if ( ERR_DB_SUCCESS != retcode )
  {
    for( jj = 0; jj < DELIVERY_RESPONSE_COUNT; jj++ )
     if( NULL != CompletedDeliveries[jj] ) {
#ifdef FFE_DEBUG
          CompletedDeliveries[jj]->iStage |= UNRESERVING;
#endif
          UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS,
CompletedDeliveries[jj] );
     }

    SendErrorResponse( pECB, ERR_DB_DELIVERY_NOT_QUEUED,
                             ERR_TYPE_WEBDLL, NULL,
                             pDelivery->w_id, pDelivery->ld_id,
                             (pConnData)pDelivery );
```

```
  return;
 }

 RESERVE_RESPONSE( DELIVERY_RESPONSE, deliveryForm );

 PutNumeric(WDID(pDelivery->w_id,pDelivery->ld_id),
                deliveryFormIndexesP[D_WDID].iLen,
                &deliveryForm[deliveryFormIndexesP[D_WDID].iStartIndex]);
 PutNumeric(pDelivery->w_id,
                deliveryFormIndexesP[D_WID].iLen,
                &deliveryForm[deliveryFormIndexesP[D_WID].iStartIndex]);
 PutNumeric(pDelivery->o_carrier_id,
                deliveryFormIndexesP[D_CAR].iLen,
                &deliveryForm[deliveryFormIndexesP[D_CAR].iStartIndex]);

 index = D_QUEUE1;
 for( jj = 0; jj < DELIVERY_RESPONSE_COUNT; jj++ ) {
  if( NULL == CompletedDeliveries[jj] )
   pCompletedDelivery = &blankDelivery;
  else
   pCompletedDelivery = CompletedDeliveries[jj];
  PutNumeric(pCompletedDelivery->queue_time,
                deliveryFormIndexesP[index].iLen,
                &deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
  index++;
  PutNumeric(pCompletedDelivery->delta_time,
                deliveryFormIndexesP[index].iLen,
                &deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
  index++;
  PutNumeric(pCompletedDelivery->w_id,
                deliveryFormIndexesP[index].iLen,
                &deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
  index++;
  PutNumeric(pCompletedDelivery->o_carrier_id,
                deliveryFormIndexesP[index].iLen,
                &deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
  index++;
  for( ii = 0; ii < 10; ii++ ) {
   PutNumeric(pCompletedDelivery->o_id[ii],
                deliveryFormIndexesP[index].iLen,

&deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
    index++;
   }
  if( NULL != CompletedDeliveries[jj] ) {
#ifdef FFE_DEBUG
    CompletedDeliveries[jj]->iStage |= UNRESERVING;
#endif
    UNRESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS,
CompletedDeliveries[jj] );
   }
 }

 PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);
 PutHTMLStrings(StrStruct, deliveryForm,
giResponseLen[DELIVERY_RESPONSE],
                &szOutput, &iOutputLen);

 SendResponse(pECB, szOutput, iOutputLen);

 UNRESERVE_RESPONSE( DELIVERY_RESPONSE, deliveryForm );

 if( szOutput != deliveryForm )
  UNRESERVE_PANIC_FORM( szOutput );
}



void
TPCCNewOrderResponse( int retcode, pNewOrderData pNewOrder )
{
 int             i;
 char  szDate[]   = "xx-xx-xxxx xx:xx:xx";
 char  szBlanks[] = "                    ";
 char  szDollar[] = "$";
 PutStrStruct StrStruct[133];
 int ssCnt = 0;
 int jj;
 int kk;
 int mm;
 char *newOrderForm;
 char *szOutput;
 int iOutputLen;
 BOOL bValid;
 char *execution_status;
 char szStatus[80];
 EXTENSION_CONTROL_BLOCK *pECB;

 pECB = pNewOrder->pCC;

 if ( ERR_DB_PENDING == retcode )
 {
  return;
 }
 else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
 {
  SendErrorResponse( pECB, ERR_NEW_ORDER_NOT_PROCESSED,
                        ERR_TYPE_WEBDLL, NULL,
                        pNewOrder->w_id, pNewOrder->ld_id,
                        (pConnData)pNewOrder );
  return;
 }
 else if( ERR_DB_SUCCESS != retcode && ERR_DB_NOT_COMMITED !=
retcode )
 {
  sprintf( szStatus,
            "Item number is not valid, or DB error = %d",
            pNewOrder->dbstatus );
  SendErrorResponse( pECB, ERR_DB_ERROR,
                        ERR_TYPE_WEBDLL, NULL,
                        pNewOrder->w_id, pNewOrder->ld_id,
                        (pConnData)pNewOrder );
  return;
 }
 else if ( ERR_DB_SUCCESS == retcode )
 {
  bValid = TRUE;
  execution_status = "Transaction commited.";
 }
 else if ( ERR_DB_NOT_COMMITED == retcode )
 {
  bValid = FALSE;
  execution_status = "Item number is not valid.";
 }

 RESERVE_RESPONSE( NEW_ORDER_RESPONSE, newOrderForm );

 if(bValid)
 {
  PutNumeric(WDID(pNewOrder->w_id,pNewOrder->ld_id),
            newOrderResponseIndexes[NO_WDID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_WDID].iStartIndex]);
  PutNumeric(pNewOrder->w_id,
            newOrderResponseIndexes[NO_WID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_WID].iStartIndex]);
  PutNumeric(pNewOrder->d_id,
            newOrderResponseIndexes[NO_DID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_DID].iStartIndex]);
```

```
   PutNumeric(pNewOrder->o_entry_d.day, 2, &szDate[0]);
   PutNumeric(pNewOrder->o_entry_d.month, 2, &szDate[3]);
   PutNumeric(pNewOrder->o_entry_d.year, 4, &szDate[6]);
   PutNumeric(pNewOrder->o_entry_d.hour, 2, &szDate[11]);
   PutNumeric(pNewOrder->o_entry_d.minute, 2, &szDate[14]);
   PutNumeric(pNewOrder->o_entry_d.second, 2, &szDate[17]);

memcpy(&newOrderForm[newOrderResponseIndexes[NO_DATE].iStartIndex],
                szDate, newOrderResponseIndexes[NO_DATE].iLen);
 }
 else
 {


memcpy(&newOrderForm[newOrderResponseIndexes[NO_DATE].iStartIndex],
                szBlanks, newOrderResponseIndexes[NO_DATE].iLen);
 }

 PutNumeric(pNewOrder->c_id,
                  newOrderResponseIndexes[NO_CID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_CID].iStartIndex]);


 PUT_STRING(pNewOrder->c_last,
                  newOrderResponseIndexes[NO_LAST].iLen,
                  newOrderResponseIndexes[NO_LAST].iStartIndex,
                  StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pNewOrder->c_credit,
                  newOrderResponseIndexes[NO_CREDIT].iLen,
                  newOrderResponseIndexes[NO_CREDIT].iStartIndex,
                  StrStruct[ssCnt]);
 ssCnt++;

 if(bValid)
 {

   PutFloat2(pNewOrder->c_discount,
                   newOrderResponseIndexes[NO_DISC].iLen,

&newOrderForm[newOrderResponseIndexes[NO_DISC].iStartIndex]);
   PutNumeric(pNewOrder->o_id,
                   newOrderResponseIndexes[NO_OID].iLen,

&newOrderForm[newOrderResponseIndexes[NO_OID].iStartIndex]);
   PutNumeric(pNewOrder->o_ol_cnt,
                   newOrderResponseIndexes[NO_LINES].iLen,

&newOrderForm[newOrderResponseIndexes[NO_LINES].iStartIndex]);
   PutFloat2(pNewOrder->w_tax,
                   newOrderResponseIndexes[NO_W_TAX].iLen,

&newOrderForm[newOrderResponseIndexes[NO_W_TAX].iStartIndex]);
   PutFloat2(pNewOrder->d_tax,
                   newOrderResponseIndexes[NO_D_TAX].iLen,

&newOrderForm[newOrderResponseIndexes[NO_D_TAX].iStartIndex]);


   for(i=0; i<pNewOrder->o_ol_cnt; i++)
   {
    PutNumeric(pNewOrder->o_ol[i].ol_supply_w_id,
                        newOrderResponseIndexes[NO_S_WID+(i*8)].iLen,

&newOrderForm[newOrderResponseIndexes[NO_S_WID+(i*8)].iStartIndex]);
     PutNumeric(pNewOrder->o_ol[i].ol_i_id,
                        newOrderResponseIndexes[NO_IID+(i*8)].iLen,

&newOrderForm[newOrderResponseIndexes[NO_IID+(i*8)].iStartIndex]);
     PUT_STRING(pNewOrder->o_ol[i].i_name,
                        newOrderResponseIndexes[NO_INAME+(i*8)].iLen,
newOrderResponseIndexes[NO_INAME+(i*8)].iStartIndex,
                        StrStruct[ssCnt]);
     ssCnt++;
     PutNumeric(pNewOrder->o_ol[i].ol_quantity,
                        newOrderResponseIndexes[NO_QTY+(i*8)].iLen,

&newOrderForm[newOrderResponseIndexes[NO_QTY+(i*8)].iStartIndex]);

     PutNumeric(pNewOrder->o_ol[i].s_quantity,
                        newOrderResponseIndexes[NO_STOCK+(i*8)].iLen,

&newOrderForm[newOrderResponseIndexes[NO_STOCK+(i*8)].iStartIndex]);
     PUT_STRING(pNewOrder->o_ol[i].b_g,
                        newOrderResponseIndexes[NO_BG+(i*8)].iLen,

newOrderResponseIndexes[NO_BG+(i*8)].iStartIndex,
                        StrStruct[ssCnt]);
     ssCnt++;

memcpy(&newOrderForm[newOrderResponseIndexes[NO_PRICE+(i*8)].iStart
Index-1],
                  szDollar, 1);
     PutFloat2(pNewOrder->o_ol[i].i_price,
                        newOrderResponseIndexes[NO_PRICE+(i*8)].iLen,

               &newOrderForm[newOrderResponseIndexes[NO_PRICE+(i*8)].iSt
artIndex]);

memcpy(&newOrderForm[newOrderResponseIndexes[NO_AMT+(i*8)].iStartIn
dex-1],
                  szDollar, 1);
     PutFloat2(pNewOrder->o_ol[i].ol_amount,
                        newOrderResponseIndexes[NO_AMT+(i*8)].iLen,

               &newOrderForm[newOrderResponseIndexes[NO_AMT+(i*8)].iSta
rtIndex]);

    }

    jj = NO_AMT + ((i-1)*8) + 1;
    for(kk=i; kk<15; kk++)
    {

     for(mm=0; mm<6; mm++)
     {
               memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartInde
x],
                    szBlanks, newOrderResponseIndexes[jj].iLen);
               jj++;
     }

     for(mm=0; mm<2; mm++)
     {
               memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartInde
x-1],
                    szBlanks, newOrderResponseIndexes[jj].iLen+1);
               jj++;
     }
    }
 }
 else
 {



memcpy(&newOrderForm[newOrderResponseIndexes[NO_DISC].iStartIndex],
                szBlanks, newOrderResponseIndexes[NO_DISC].iLen);

   PutNumeric(pNewOrder->o_id,
                   newOrderResponseIndexes[NO_OID].iLen,
```

```
&newOrderForm[newOrderResponseIndexes[NO_OID].iStartIndex]);

   for(kk=0; kk<3; kk++)
   {

memcpy(&newOrderForm[newOrderResponseIndexes[NO_LINES+kk].iStartIn
dex],
                  szBlanks, newOrderResponseIndexes[NO_LINES+kk].iLen);
   }

   jj = NO_S_WID;
   for(kk=0; kk<15; kk++)
   {

    for(mm=0; mm<6; mm++)
    {
              memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartInde
x],
                   szBlanks, newOrderResponseIndexes[jj].iLen);
              jj++;
    }

    for(mm=0; mm<2; mm++)
    {
              memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStartInde
x-1],
                   szBlanks, newOrderResponseIndexes[jj].iLen+1);
              jj++;
    }

   }
  }


  PUT_STRING(execution_status, newOrderResponseIndexes[NO_STAT].iLen,
                  newOrderResponseIndexes[NO_STAT].iStartIndex,
                  StrStruct[ssCnt]);
  ssCnt++;

  if(bValid)
  {

   PutFloat2(pNewOrder->total_amount,
                  newOrderResponseIndexes[NO_TOTAL].iLen,

&newOrderForm[newOrderResponseIndexes[NO_TOTAL].iStartIndex]);
  }
  else
  {


memcpy(&newOrderForm[newOrderResponseIndexes[NO_TOTAL].iStartIndex
],
                szBlanks, newOrderResponseIndexes[NO_TOTAL].iLen);
  }
  PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);
  PutHTMLStrings(StrStruct, newOrderForm,
giResponseLen[NEW_ORDER_RESPONSE],
                           &szOutput, &iOutputLen);

#ifdef FFE_DEBUG
  pNewOrder->iStage |= UNRESERVING;
#endif

  UNRESERVE_TRANSACTION_STRUCT( NEW_ORDER_TRANS,
pNewOrder );

  SendResponse(pECB, szOutput, iOutputLen);

  UNRESERVE_RESPONSE( NEW_ORDER_RESPONSE, newOrderForm );

  if( szOutput != newOrderForm )
    UNRESERVE_PANIC_FORM( szOutput );
}



void
TPCCPaymentResponse( int retcode, pPaymentData pPayment )
{
  char          *ptr;
  char          szTmp[64];
  char          szW_Zip[26];
  char          szD_Zip[26];
  char          szC_Zip[26];
  char          szC_Phone[26];
  int           i;
  int           l;
  char          *szZipPic = "XXXXX-XXXX";
  char  szLongDate[] = "XX-XX-XXXX XX:XX:XX";
  char  szDate[]   = "xx-xx-xxxx";
  char  szBlanks[] = "                                        ";
  PutStrStruct StrStruct[34];
  int ssCnt = 0;
  char *paymentForm;
  char *szOutput;
  int iOutputLen;
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = pPayment->pCC;

  if ( ERR_DB_PENDING == retcode )
  {
   return;
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
   SendErrorResponse( pECB, ERR_PAYMENT_NOT_PROCESSED,
                           ERR_TYPE_WEBDLL, NULL,
                           pPayment->w_id, pPayment->ld_id,
                           (pConnData)pPayment );
   return;
  }
  else if ( ERR_DB_NOT_COMMITED == retcode )
  {
   SendErrorResponse( pECB, ERR_PAYMENT_INVALID_CUSTOMER,
                           ERR_TYPE_WEBDLL, NULL,
                           pPayment->w_id, pPayment->ld_id,
                           (pConnData)pPayment );
   return;
  }
  else if ( ERR_DB_SUCCESS != retcode )
  {
   SendErrorResponse( pECB, ERR_DB_ERROR,
                           ERR_TYPE_WEBDLL, NULL,
                           pPayment->w_id, pPayment->ld_id,
                           (pConnData)pPayment );
   return;
  }

  RESERVE_RESPONSE( PAYMENT_RESPONSE,  paymentForm );

  PutNumeric(WDID(pPayment->w_id,pPayment->ld_id),
                  paymentResponseIndexes[PT_WDID].iLen,

&paymentForm[paymentResponseIndexes[PT_WDID].iStartIndex]);
  PutNumeric(pPayment->h_date.day, 2,
                  &szLongDate[0]);
  PutNumeric(pPayment->h_date.month, 2,
                  &szLongDate[3]);
  PutNumeric(pPayment->h_date.year, 4,
                  &szLongDate[6]);
  PutNumeric(pPayment->h_date.hour, 2,
```

```
                &szLongDate[11]);
 PutNumeric(pPayment->h_date.minute, 2,
                &szLongDate[14]);
 PutNumeric(pPayment->h_date.second, 2,
                &szLongDate[17]);

memcpy(&paymentForm[paymentResponseIndexes[PT_LONG_DATE].iStartIn
dex],
              szLongDate, paymentResponseIndexes[PT_LONG_DATE].iLen);

 PutNumeric(pPayment->w_id,
                paymentResponseIndexes[PT_WID].iLen,

&paymentForm[paymentResponseIndexes[PT_WID].iStartIndex]);
 PutNumeric(pPayment->d_id,
                paymentResponseIndexes[PT_DID].iLen,

&paymentForm[paymentResponseIndexes[PT_DID].iStartIndex]);

 PUT_STRING(pPayment->w_street_1,
                paymentResponseIndexes[PT_W_ST_1].iLen,
                paymentResponseIndexes[PT_W_ST_1].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->d_street_1,
                paymentResponseIndexes[PT_D_ST_1].iLen,
                paymentResponseIndexes[PT_D_ST_1].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->w_street_2,
                paymentResponseIndexes[PT_W_ST_2].iLen,
                paymentResponseIndexes[PT_W_ST_2].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->d_street_2,
                paymentResponseIndexes[PT_D_ST_2].iLen,
                paymentResponseIndexes[PT_D_ST_2].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->w_city,
                paymentResponseIndexes[PT_W_CITY].iLen,
                paymentResponseIndexes[PT_W_CITY].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->w_state,
                paymentResponseIndexes[PT_W_ST].iLen,
                paymentResponseIndexes[PT_W_ST].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 FormatString(szW_Zip, szZipPic, pPayment->w_zip);
 memcpy(&paymentForm[paymentResponseIndexes[PT_W_ZIP].iStartIndex],
              szW_Zip, paymentResponseIndexes[PT_W_ZIP].iLen);
 PUT_STRING(pPayment->d_city,
                paymentResponseIndexes[PT_D_CITY].iLen,
                paymentResponseIndexes[PT_D_CITY].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->d_state,
                paymentResponseIndexes[PT_D_ST].iLen,
                paymentResponseIndexes[PT_D_ST].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 FormatString(szD_Zip, szZipPic, pPayment->d_zip);
 memcpy(&paymentForm[paymentResponseIndexes[PT_D_ZIP].iStartIndex],
              szD_Zip, paymentResponseIndexes[PT_D_ZIP].iLen);
 PutNumeric(pPayment->c_id,
                paymentResponseIndexes[PT_CID].iLen,

&paymentForm[paymentResponseIndexes[PT_CID].iStartIndex]);
 PutNumeric(pPayment->c_w_id,
                paymentResponseIndexes[PT_C_WID].iLen,

&paymentForm[paymentResponseIndexes[PT_C_WID].iStartIndex]);
 PutNumeric(pPayment->c_d_id,
                paymentResponseIndexes[PT_C_DID].iLen,

&paymentForm[paymentResponseIndexes[PT_C_DID].iStartIndex]);

 PUT_STRING(pPayment->c_first,
                paymentResponseIndexes[PT_FIRST].iLen,
                paymentResponseIndexes[PT_FIRST].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->c_middle,
                paymentResponseIndexes[PT_MIDDLE].iLen,
                paymentResponseIndexes[PT_MIDDLE].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->c_last,
                paymentResponseIndexes[PT_LAST].iLen,
                paymentResponseIndexes[PT_LAST].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;

 PutNumeric(pPayment->c_since.day, 2, &szDate[0]);
 PutNumeric(pPayment->c_since.month, 2, &szDate[3]);
 PutNumeric(pPayment->c_since.year, 4, &szDate[6]);

memcpy(&paymentForm[paymentResponseIndexes[PT_SM_DATE].iStartIndex
], szDate,
              paymentResponseIndexes[PT_SM_DATE].iLen);

 PUT_STRING(pPayment->c_street_1,
                paymentResponseIndexes[PT_C_STR_1].iLen,
                paymentResponseIndexes[PT_C_STR_1].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pPayment->c_credit,
                paymentResponseIndexes[PT_CREDIT].iLen,
                paymentResponseIndexes[PT_CREDIT].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;

 PUT_STRING(pPayment->d_street_2,
                paymentResponseIndexes[PT_D_STR_2].iLen,
                paymentResponseIndexes[PT_D_STR_2].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;

 PutFloat2(pPayment->c_discount,
               paymentResponseIndexes[PT_DISC].iLen,

&paymentForm[paymentResponseIndexes[PT_DISC].iStartIndex]);

 PUT_STRING(pPayment->c_city,
                paymentResponseIndexes[PT_C_CITY].iLen,
                paymentResponseIndexes[PT_C_CITY].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;

 PUT_STRING(pPayment->c_state,
                paymentResponseIndexes[PT_C_ST].iLen,
                paymentResponseIndexes[PT_C_ST].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;

 FormatString(szC_Zip, szZipPic, pPayment->c_zip);
 memcpy(&paymentForm[paymentResponseIndexes[PT_C_ZIP].iStartIndex],
szC_Zip,
              paymentResponseIndexes[PT_C_ZIP].iLen);
 FormatString(szC_Phone, "XXXXXX-XXX-XXX-XXXX",
                pPayment->c_phone);
```

```
memcpy(&paymentForm[paymentResponseIndexes[PT_C_PHONE].iStartIndex
],
                szC_Phone, paymentResponseIndexes[PT_C_PHONE].iLen);

 PutFloat2(pPayment->h_amount,
                paymentResponseIndexes[PT_AMT].iLen,

&paymentForm[paymentResponseIndexes[PT_AMT].iStartIndex]);
 PutFloat2(pPayment->c_balance,
                paymentResponseIndexes[PT_BAL].iLen,

&paymentForm[paymentResponseIndexes[PT_BAL].iStartIndex]);

 PutFloat2(pPayment->c_credit_lim,
                paymentResponseIndexes[PT_LIM].iLen,

&paymentForm[paymentResponseIndexes[PT_LIM].iStartIndex]);

 ptr = pPayment->c_credit;
 if ( *ptr == 'B' && *(ptr+1) == 'C' )
 {
  ptr = pPayment->c_data;
  l = strlen( ptr ) / 50;
  for(i=0; i<4; i++, ptr += 50)
  {
   if ( i <= l )
   {
                strncpy(szTmp, ptr, 50);
                szTmp[50] = '\0';
   }
   else
                szTmp[0] = 0;

   PUT_STRING(szTmp,
                        paymentResponseIndexes[PT_CUST_DATA+i].iLen,

paymentResponseIndexes[PT_CUST_DATA+i].iStartIndex,
                        StrStruct[ssCnt]);
   ssCnt++;
  }
 }
 else
 {
  for(i=0; i<4; i++)
  {

memcpy(&paymentForm[paymentResponseIndexes[PT_CUST_DATA+i].iStartI
ndex],
                szBlanks, paymentResponseIndexes[PT_CUST_DATA+i].iLen);
  }
 }

 PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);

 PutHTMLStrings(StrStruct, paymentForm,
giResponseLen[PAYMENT_RESPONSE],
                        &szOutput, &iOutputLen);

#ifdef FFE_DEBUG
 pPayment->iStage |= UNRESERVING;
#endif

 UNRESERVE_TRANSACTION_STRUCT( PAYMENT_TRANS, pPayment );

 SendResponse(pECB, szOutput, iOutputLen);

 UNRESERVE_RESPONSE( PAYMENT_RESPONSE, paymentForm );

 if( szOutput != paymentForm )
  UNRESERVE_PANIC_FORM( szOutput );
}
```

```
void
TPCCOrderStatusResponse( int retcode, pOrderStatusData pOrderStatus )
{
 int          i;
 int          jj;
 int          kk;
 int          mm;
 char  szLongDate[] = "XX-XX-XXXX XX:XX:XX";
 char  szDate[]   = "XX-XX-XXXX";
 char  szBlanks[] = "                    ";
 char  szDollar[] = "$";
 PutStrStruct StrStruct[4];
 int   ssCnt = 0;
 char         *orderStatusForm;
 char         *szOutput;
 int          iOutputLen;
 EXTENSION_CONTROL_BLOCK *pECB;

 pECB = pOrderStatus->pCC;

 if ( ERR_DB_PENDING == retcode )
 {
  return;
 }
 else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
 {
  SendErrorResponse( pECB, ERR_ORDER_STATUS_NOT_PROCESSED,
                           ERR_TYPE_WEBDLL, NULL,
                           pOrderStatus->w_id, pOrderStatus->ld_id,
                           (pConnData)pOrderStatus );
  return;
 }
 else if ( ERR_DB_NOT_COMMITED == retcode )
 {
  SendErrorResponse( pECB, ERR_NOSUCH_CUSTOMER,
                           ERR_TYPE_WEBDLL, NULL,
                           pOrderStatus->w_id, pOrderStatus->ld_id,
                           (pConnData)pOrderStatus );
  return;
 }
 else         if ( ERR_DB_SUCCESS != retcode )
 {
  SendErrorResponse( pECB, ERR_DB_ERROR,
                           ERR_TYPE_WEBDLL, NULL,
                           pOrderStatus->w_id, pOrderStatus->ld_id,
                           (pConnData)pOrderStatus );
 return;
 }

 RESERVE_RESPONSE( ORDER_STATUS_RESPONSE, orderStatusForm );

 PutNumeric(WDID(pOrderStatus->w_id,pOrderStatus->ld_id),
                orderStatusResponseIndexes[OS_WDID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_WDID].iStartIndex]);
 PutNumeric(pOrderStatus->w_id,
                orderStatusResponseIndexes[OS_WID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_WID].iStartIndex]);
 PutNumeric(pOrderStatus->d_id,
                orderStatusResponseIndexes[OS_DID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_DID].iStartIndex]);
 PutNumeric(pOrderStatus->c_id,
                orderStatusResponseIndexes[OS_CID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_CID].iStartIndex]);
 PUT_STRING(pOrderStatus->c_first,
                orderStatusResponseIndexes[OS_FIRST].iLen,
```

```
                orderStatusResponseIndexes[OS_FIRST].iStartIndex,
StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pOrderStatus->c_middle,
                orderStatusResponseIndexes[OS_MIDDLE].iLen,
                orderStatusResponseIndexes[OS_MIDDLE].iStartIndex,
                StrStruct[ssCnt]);
 ssCnt++;
 PUT_STRING(pOrderStatus->c_last,
                orderStatusResponseIndexes[OS_LAST].iLen,
                orderStatusResponseIndexes[OS_LAST].iStartIndex,
StrStruct[ssCnt]);
 ssCnt++;
 PutFloat2(pOrderStatus->c_balance,
               orderStatusResponseIndexes[OS_BAL].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_BAL].iStartIndex]);
 PutNumeric(pOrderStatus->o_id,
                orderStatusResponseIndexes[OS_OID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_OID].iStartIndex]);

 PutNumeric(pOrderStatus->o_entry_d.day, 2, &szLongDate[0]);
 PutNumeric(pOrderStatus->o_entry_d.month, 2, &szLongDate[3]);
 PutNumeric(pOrderStatus->o_entry_d.year, 4, &szLongDate[6]);
 PutNumeric(pOrderStatus->o_entry_d.hour, 2, &szLongDate[11]);
 PutNumeric(pOrderStatus->o_entry_d.minute, 2, &szLongDate[14]);
 PutNumeric(pOrderStatus->o_entry_d.second, 2, &szLongDate[17]);

memcpy(&orderStatusForm[orderStatusResponseIndexes[OS_DATE].iStartInde
x],
            szLongDate, orderStatusResponseIndexes[OS_DATE].iLen);
 PutNumeric(pOrderStatus->o_carrier_id,
                orderStatusResponseIndexes[OS_CAR_ID].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_CAR_ID].iStartIndex]);

 for(i=0; i<pOrderStatus->o_ol_cnt; i++)
 {
  PutNumeric(pOrderStatus->s_ol[i].ol_supply_w_id,
                orderStatusResponseIndexes[OS_S_WID+(i*5)].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_S_WID+(i*5)].iStartIndex]
);
  PutNumeric(pOrderStatus->s_ol[i].ol_i_id,
                orderStatusResponseIndexes[OS_IID+(i*5)].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_IID+(i*5)].iStartIndex]);
  PutNumeric(pOrderStatus->s_ol[i].ol_quantity,
                orderStatusResponseIndexes[OS_QTY+(i*5)].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_QTY+(i*5)].iStartIndex]);

memcpy(&orderStatusForm[orderStatusResponseIndexes[OS_AMT+(i*5)].iStar
tIndex-1],
                szDollar, 1);
  PutFloat2(pOrderStatus->s_ol[i].ol_amount,
                orderStatusResponseIndexes[OS_AMT+(i*5)].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_AMT+(i*5)].iStartIndex]);
  PutNumeric(pOrderStatus->s_ol[i].ol_delivery_d.day,
                2, &szDate[0]);
  PutNumeric(pOrderStatus->s_ol[i].ol_delivery_d.month,
                2, &szDate[3]);
  PutNumeric(pOrderStatus->s_ol[i].ol_delivery_d.year,
                4, &szDate[6]);

memcpy(&orderStatusForm[orderStatusResponseIndexes[OS_SM_DATE+(i*5)
].iStartIndex],
               szDate,
orderStatusResponseIndexes[OS_SM_DATE+(i*5)].iLen);
 }

 jj = OS_SM_DATE + ((i-1)*5) + 1;
 for(kk=i; kk<15; kk++)
 {

  for(mm=0; mm<3; mm++)
  {
   memcpy(&orderStatusForm[orderStatusResponseIndexes[jj].iStartIndex],
                szBlanks, orderStatusResponseIndexes[jj].iLen);
   jj++;
  }

  memcpy(&orderStatusForm[orderStatusResponseIndexes[jj].iStartIndex-1],
                szBlanks, orderStatusResponseIndexes[jj].iLen+1);
  jj++;
  memcpy(&orderStatusForm[orderStatusResponseIndexes[jj].iStartIndex],
                szBlanks, orderStatusResponseIndexes[jj].iLen);
  jj++;
 }

 PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);
 PutHTMLStrings(StrStruct, orderStatusForm,
                            giResponseLen[ORDER_STATUS_RESPONSE],
                            &szOutput, &iOutputLen);

#ifdef FFE_DEBUG
 pOrderStatus->iStage |= UNRESERVING;
#endif

 UNRESERVE_TRANSACTION_STRUCT( ORDER_STATUS_TRANS,
pOrderStatus );

 SendResponse(pECB, szOutput, iOutputLen);

 UNRESERVE_RESPONSE( ORDER_STATUS_RESPONSE, orderStatusForm
);

 if( szOutput != orderStatusForm )
  UNRESERVE_PANIC_FORM( szOutput );
}



void
TPCCStockLevelResponse( int retcode, StockLevelData *pStockLevel )
{
 char *stockLevelForm;
 EXTENSION_CONTROL_BLOCK *pECB;

 pECB = pStockLevel->pCC;

 if ( ERR_DB_PENDING == retcode )
 {
  return;
 }
 else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
 {
  SendErrorResponse( pECB, ERR_STOCKLEVEL_NOT_PROCESSED,
                           ERR_TYPE_WEBDLL, NULL,
                           pStockLevel->w_id, pStockLevel->ld_id,
                           (pConnData)pStockLevel );
  return;
 }
 else if ( ERR_DB_SUCCESS != retcode )
 {
  SendErrorResponse( pECB, ERR_DB_ERROR,
                           ERR_TYPE_WEBDLL, NULL,
                           pStockLevel->w_id, pStockLevel->ld_id,
                           (pConnData)pStockLevel );
  return;
 }
```

```
  RESERVE_RESPONSE( STOCK_LEVEL_RESPONSE, stockLevelForm );

  PutNumeric(WDID(pStockLevel->w_id,pStockLevel->ld_id),
                    stockLevelResponseIndexes[SL_WDID].iLen,
&stockLevelForm[stockLevelResponseIndexes[SL_WDID].iStartIndex]);
  PutNumeric(pStockLevel->w_id,
                    stockLevelResponseIndexes[SL_WID].iLen,
&stockLevelForm[stockLevelResponseIndexes[SL_WID].iStartIndex]);
  PutNumeric(pStockLevel->ld_id,
                    stockLevelResponseIndexes[SL_DID].iLen,
&stockLevelForm[stockLevelResponseIndexes[SL_DID].iStartIndex]);
  PutNumeric(pStockLevel->threshold,
                    stockLevelResponseIndexes[SL_TH].iLen,
&stockLevelForm[stockLevelResponseIndexes[SL_TH].iStartIndex]);
  PutNumeric(pStockLevel->low_stock,
                    stockLevelResponseIndexes[SL_LOW].iLen,
&stockLevelForm[stockLevelResponseIndexes[SL_LOW].iStartIndex]);

#ifdef FFE_DEBUG
  pStockLevel->iStage |= UNRESERVING;
#endif

  UNRESERVE_TRANSACTION_STRUCT( STOCK_LEVEL_TRANS,
pStockLevel );

  SendResponse(pECB, stockLevelForm,
                    giResponseLen[STOCK_LEVEL_RESPONSE]);

  UNRESERVE_RESPONSE( STOCK_LEVEL_RESPONSE, stockLevelForm );
}



void
TPCCResponseComplete( EXTENSION_CONTROL_BLOCK *pECB )
{
  DWORD    status;

  status = HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  (pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_DONE_WITH_SESSION,
                                          &status, NULL, NULL);
}



int
ProcessDeliveryQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                              int w_id, int ld_id )
{
  int                              retcode;
  char                             *ptr;
  char                             *deliveryVals[MAXDELIVERYVALS];
  pDeliveryData                    pDelivery;
  pDeliveryData
              CompletedDeliveries[DELIVERY_RESPONSE_COUNT];

  RESERVE_TRANSACTION_STRUCT( DELIVERY_TRANS, pDelivery );

  pDelivery->w_id = w_id;
  pDelivery->ld_id = ld_id;
  pDelivery->pCC = pECB;

  PARSE_QUERY_STRING(lpszQueryString, MAXDELIVERYVALS,
                              deliveryStrs, deliveryVals);

  if ( !GetValuePtr(deliveryVals, QUEUETIME, &ptr) )
    return ERR_DELIVERY_MISSING_QUEUETIME_KEY;

  if ( !GetNumeric(ptr, &pDelivery->queue_time) )
    return ERR_DELIVERY_QUEUETIME_INVALID;

  if ( !GetValuePtr(deliveryVals, OCD, &ptr) )
    return ERR_DELIVERY_MISSING_OCD_KEY;

  if ( !GetNumeric(ptr, &pDelivery->o_carrier_id) )
    return ERR_DELIVERY_CARRIER_INVALID;

  if ( pDelivery->o_carrier_id > 10 || pDelivery->o_carrier_id < 1 )
    return ERR_DELIVERY_CARRIER_ID_RANGE;

#ifdef FFE_DEBUG
  pDelivery->iStage |= CALLING_LH;
#endif
  retcode = TPCCDelivery( pDelivery, CompletedDeliveries );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
  pDelivery->iStage |= CALLING_RESP;
#endif
  TPCCDeliveryResponse( retcode, pDelivery, CompletedDeliveries );

  return retcode;
}



int
ProcessNewOrderQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                              int w_id, int ld_id )
{
  int                              retcode;
  NewOrderData                     *pNewOrder;

  RESERVE_TRANSACTION_STRUCT( NEW_ORDER_TRANS, pNewOrder
);

  pNewOrder->w_id = w_id;
  pNewOrder->ld_id = ld_id;
  pNewOrder->pCC = pECB;

  if ( ERR_SUCCESS != ( retcode = ParseNewOrderQuery( lpszQueryString,

pNewOrder )))
    return retcode;

#ifdef FFE_DEBUG
  pNewOrder->iStage |= CALLING_LH;
#endif
  retcode = TPCCNewOrder( pNewOrder );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
  pNewOrder->iStage |= CALLING_RESP;
#endif
  TPCCNewOrderResponse( retcode, pNewOrder );

  return retcode;
}



int
ProcessOrderStatusQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                              int w_id, int ld_id )
```

```
{
 int                                    retcode;
 OrderStatusData          *pOrderStatus;

 RESERVE_TRANSACTION_STRUCT( ORDER_STATUS_TRANS,
pOrderStatus );

 pOrderStatus->w_id = w_id;
 pOrderStatus->ld_id = ld_id;
 pOrderStatus->pCC = pECB;

 if( ERR_SUCCESS != ( retcode = ParseOrderStatusQuery( lpszQueryString,

              pOrderStatus )))
   return retcode;

#ifdef FFE_DEBUG
 pOrderStatus->iStage |= CALLING_LH;
#endif
 retcode = TPCCOrderStatus( pOrderStatus );

#ifdef FFE_DEBUG
 _ASSERT(VALID_DB_ERR(retcode));
 pOrderStatus->iStage |= CALLING_RESP;
#endif
 TPCCOrderStatusResponse( retcode, pOrderStatus );

 return retcode;
}



int
ProcessPaymentQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                              int w_id, int ld_id )
{
 int                                    retcode;
 PaymentData                            *pPayment;

 RESERVE_TRANSACTION_STRUCT( PAYMENT_TRANS, pPayment );

 pPayment->w_id = w_id;
 pPayment->ld_id = ld_id;
 pPayment->pCC = pECB;

 if( ERR_SUCCESS != ( retcode = ParsePaymentQuery( lpszQueryString,

pPayment )))
   return retcode;

#ifdef FFE_DEBUG
 pPayment->iStage |= CALLING_LH;
#endif
 retcode = TPCCPayment( pPayment );

#ifdef FFE_DEBUG
 _ASSERT(VALID_DB_ERR(retcode));
 pPayment->iStage |= CALLING_RESP;
#endif
 TPCCPaymentResponse( retcode, pPayment );

 return retcode;
}



int
ProcessStockLevelQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                                    int w_id, int ld_id )
{
 char        *ptr;
 int                    retcode;
 char                   *stockLevelVals[MAXSTOCKLEVELVALS];
 StockLevelData         *pStockLevel;

 RESERVE_TRANSACTION_STRUCT( STOCK_LEVEL_TRANS,
pStockLevel );

 pStockLevel->w_id = w_id;
 pStockLevel->ld_id = ld_id;
 pStockLevel->pCC = pECB;

 PARSE_QUERY_STRING(lpszQueryString, MAXSTOCKLEVELVALS,
                        stockLevelStrs, stockLevelVals);

 if ( !GetValuePtr(stockLevelVals, TT, &ptr))
  return ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY;

 if ( !GetNumeric(ptr, &pStockLevel->threshold) )
  return ERR_STOCKLEVEL_THRESHOLD_INVALID;

 if ( pStockLevel->threshold >= 100 || pStockLevel->threshold < 0 )
  return ERR_STOCKLEVEL_THRESHOLD_RANGE;

#ifdef FFE_DEBUG
 pStockLevel->iStage |= CALLING_LH;
#endif
 retcode = TPCCStockLevel( pStockLevel );

#ifdef FFE_DEBUG
 _ASSERT(VALID_DB_ERR(retcode));
 pStockLevel->iStage |= CALLING_RESP;
#endif
 TPCCStockLevelResponse( retcode, pStockLevel );

 return retcode;
}



BOOL
GetValuePtr(char *pProcessedQuery[], int iIndex, char **pValue)
{
 *pValue = pProcessedQuery[iIndex];

 if(NULL == *pValue)return FALSE;

 return TRUE;
}



void
MakeDeliveryTemplates( char *deliveryForm, char *deliveryResponse )
{
 int            curLen;


 curLen = sprintf(deliveryForm, szFormTemplate, szModName);
 ParseTemplateString(deliveryForm, &curLen, szDeliveryFormTemp2i,
                              deliveryFormIndexesI);
 giFormLen[DELIVERY_FORM] = curLen;


 curLen = sprintf(deliveryResponse, szFormTemplate, szModName);
 ParseTemplateString(deliveryResponse, &curLen, szDeliveryFormTemp2p,
                              deliveryFormIndexesP);
 giResponseLen[DELIVERY_RESPONSE] = curLen;
}
```

```
void
MakeNewOrderTemplates( char *newOrderForm, char *newOrderResponse )
{
 int            curLen;


 curLen = sprintf(newOrderForm, szFormTemplate, szModName);
 ParseTemplateString(newOrderForm, &curLen, szNewOrderFormTemp2i,
                                 newOrderFormIndexes);
 giFormLen[NEW_ORDER_FORM] = curLen;


 curLen = sprintf(newOrderResponse, szFormTemplate, szModName);
 ParseTemplateString(newOrderResponse, &curLen, szNewOrderFormTemp2p,
                                 newOrderResponseIndexes);
 giResponseLen[NEW_ORDER_RESPONSE] = curLen;
}



void
MakeOrderStatusTemplates(char *orderStatusForm, char *orderStatusResponse)
{
 int            curLen;


 curLen = sprintf(orderStatusForm, szFormTemplate, szModName);
 ParseTemplateString(orderStatusForm, &curLen, szOrderStatusFormTemp2i,
                                 orderStatusFormIndexes);
 giFormLen[ORDER_STATUS_FORM] = curLen;


 curLen = sprintf(orderStatusResponse, szFormTemplate, szModName);
 ParseTemplateString(orderStatusResponse, &curLen,
szOrderStatusFormTemp2p,
                                 orderStatusResponseIndexes);
 giResponseLen[ORDER_STATUS_RESPONSE] = curLen;
}



void
MakePaymentTemplates(char *paymentForm, char *paymentResponse)
{
 int            curLen;


 curLen = sprintf(paymentForm, szFormTemplate, szModName);
 ParseTemplateString(paymentForm, &curLen, szPaymentFormTemp2i,
                                 paymentFormIndexes);
 giFormLen[PAYMENT_FORM] = curLen;


 curLen = sprintf(paymentResponse, szFormTemplate, szModName);
 ParseTemplateString(paymentResponse, &curLen, szPaymentFormTemp2p,
                                 paymentResponseIndexes);
 giResponseLen[PAYMENT_RESPONSE] = curLen;
}


void
MakeStockLevelTemplates(char *stockLevelForm, char *stockLevelResponse)
{
 int            curLen;


 curLen = sprintf(stockLevelForm, szFormTemplate, szModName);
 ParseTemplateString(stockLevelForm, &curLen, szStockLevelFormTemp2i,
                                 stockLevelFormIndexes);
 giFormLen[STOCK_LEVEL_FORM] = curLen;


 curLen = sprintf(stockLevelResponse, szFormTemplate, szModName);
 ParseTemplateString(stockLevelResponse, &curLen,
szStockLevelFormTemp2p,
                                 stockLevelResponseIndexes);
 giResponseLen[STOCK_LEVEL_RESPONSE] = curLen;
}


void
MakeResponseHeader(void)
{
 ParseTemplateString(szResponseHeader, &responseHeaderLen,
                                 szResponseHeaderTemplate,
responseHeaderIndexes);
}



void
MakePanicPool( DWORD dwResponseSize )
{
 int iMallocSize;
 char *pForm;
 DWORD ii;


 iMallocSize = (((char *)&gpPanicForms->index - (char *)gpPanicForms) +
                                 (((char *)gpPanicForms->forms - (char
*)gpPanicForms->index)
                                  * dwResponseSize) +
                                 (((char *)&gpPanicForms-
>forms[PANIC_FORM_SIZE] -
                                  (char *)&gpPanicForms->forms[0]) *
dwResponseSize));
 gpPanicForms = malloc( iMallocSize );
 InitializeCriticalSection( &gpPanicForms->critSec );
#ifdef FFE_DEBUG
 gpPanicForms->iMaxIndex = dwResponseSize - 1;
#endif
 gpPanicForms->iNextFree = 0;
 pForm =
  ((char *)&gpPanicForms->index[0] +
   (((char *)&gpPanicForms->forms[0] - (char *)&gpPanicForms->index[0]) *
    dwResponseSize));

 for( ii = 0; ii < dwResponseSize; ii++ )
 {
  gpPanicForms->index[ii] = pForm;
  pForm += PANIC_FORM_SIZE;
 }
}


void
DeletePanicPool( void )
{
 DeleteCriticalSection( &gpPanicForms->critSec );
 free( gpPanicForms );
}



void
MakeTemplatePool( DWORD dwFormSize, DWORD dwResponseSize )
{
 char szDeliveryForm[sizeof(szFormTemplate)+FILENAMESIZE+
                                 sizeof(szDeliveryFormTemp2i)];
 char szNewOrderForm[sizeof(szFormTemplate)+FILENAMESIZE+
                                 sizeof(szNewOrderFormTemp2i)];
 char szOrderStatusForm[sizeof(szFormTemplate)+FILENAMESIZE+
                                         sizeof(szOrderStatusFormTemp2i)];
 char szPaymentForm[sizeof(szFormTemplate)+FILENAMESIZE+
```

```
                                sizeof(szPaymentFormTemp2i)];
  char szStockLevelForm[sizeof(szFormTemplate)+FILENAMESIZE+
                                sizeof(szStockLevelFormTemp2i)];
  char szDeliveryResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                                sizeof(szDeliveryFormTemp2p)];
  char szNewOrderResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                                sizeof(szNewOrderFormTemp2p)];
  char szOrderStatusResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                                sizeof(szOrderStatusFormTemp2p)];
  char szPaymentResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                                sizeof(szPaymentFormTemp2p)];
  char szStockLevelResponse[sizeof(szFormTemplate)+FILENAMESIZE+
                                sizeof(szStockLevelFormTemp2p)];
  int           iFormLen[NUMBER_POOL_FORM_TYPES];
  int           iResponseLen[NUMBER_POOL_RESPONSE_TYPES];
  int iMallocSize;
  int iRowSize;
  DWORD ii;
  int jj;
  char *pForm;
  char *pResponse;


  MakeDeliveryTemplates( szDeliveryForm, szDeliveryResponse );
  MakeNewOrderTemplates( szNewOrderForm, szNewOrderResponse );
  MakeOrderStatusTemplates( szOrderStatusForm, szOrderStatusResponse );
  MakePaymentTemplates( szPaymentForm, szPaymentResponse );
  MakeStockLevelTemplates( szStockLevelForm, szStockLevelResponse );
  MakeResponseHeader( );


  iRowSize = 0;
  for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
  {
   iFormLen[jj] = ( giFormLen[jj] + 8 ) & ( ~(int)7 );
   iRowSize += iFormLen[jj];
  }

  iMallocSize = (((char *)&gpForms->index - (char *)gpForms) +
                        (((char *)gpForms->forms - (char *)gpForms->index)
                         * dwFormSize * NUMBER_POOL_FORM_TYPES )
+
                        (((char *)&gpForms->forms[iRowSize * dwFormSize]
-
                         (char *)&gpForms->forms[0])));
  gpForms = malloc( iMallocSize );

  for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
  {
   InitializeCriticalSection( &gpForms->critSec[jj] );
   gpForms->iNextFreeForm[jj] = 0;
   gpForms->iFirstFormIndex[jj] = jj * dwFormSize;
#ifdef FFE_DEBUG
   gpForms->iMaxIndex[jj] = dwFormSize - 1;
#endif
  }

  pForm = ((char *)&gpForms->index[0] +
              (((char *)&gpForms->forms[0] - (char *)&gpForms->index[0]) *
               NUMBER_POOL_FORM_TYPES * dwFormSize));
  for( ii = 0; ii < dwFormSize; ii++ )
  {
   for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
   {
    gpForms->index[jj*dwFormSize+ii] = pForm;
    pForm += iFormLen[jj];
   }
  }


  pForm = gpForms->index[0];
  memcpy( pForm, szDeliveryForm, iFormLen[DELIVERY_FORM] );
  pForm += iFormLen[DELIVERY_FORM];

  memcpy( pForm, szNewOrderForm, iFormLen[NEW_ORDER_FORM] );
  pForm += iFormLen[NEW_ORDER_FORM];

  memcpy( pForm, szOrderStatusForm, iFormLen[ORDER_STATUS_FORM] );
  pForm += iFormLen[ORDER_STATUS_FORM];

  memcpy( pForm, szPaymentForm, iFormLen[PAYMENT_FORM] );
  pForm += iFormLen[PAYMENT_FORM];

  memcpy( pForm, szStockLevelForm, iFormLen[STOCK_LEVEL_FORM] );
  pForm += iFormLen[STOCK_LEVEL_FORM];


  pForm = gpForms->index[0];
  for( ii = 1; ii < dwFormSize; ii++ )
  {
   memcpy( gpForms->index[ii], pForm, iRowSize );
  }


  iRowSize = 0;
  for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
  {
   iResponseLen[jj] = ( giResponseLen[jj] + 8 ) & ( ~(int)7 );
   iRowSize += iResponseLen[jj];
  }

  iMallocSize = (((char *)&gpResponses->index - (char *)gpResponses) +
                        (((char *)gpResponses->responses - (char
*)gpResponses->index)
                         * dwResponseSize *
NUMBER_POOL_RESPONSE_TYPES ) +
                        (((char *)&gpResponses->responses[iRowSize *
dwResponseSize] -
                         (char *)&gpResponses->responses[0])));
  gpResponses = malloc( iMallocSize );

  for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
  {
   InitializeCriticalSection( &gpResponses->critSec[jj] );
#ifdef FFE_DEBUG
   gpResponses->iMaxIndex[jj] = dwResponseSize - 1;
#endif
   gpResponses->iNextFreeResponse[jj] = 0;
   gpResponses->iFirstResponseIndex[jj] = jj * dwResponseSize;
  }

  pResponse = ((char *)&gpResponses->index[0] +
              (((char *)&gpResponses->responses[0] -
               (char *)&gpResponses->index[0]) *
               NUMBER_POOL_RESPONSE_TYPES * dwResponseSize));
  for( ii = 0; ii < dwResponseSize; ii++ )
  {
   for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
   {
    gpResponses->index[jj*dwResponseSize+ii] = pResponse;
    pResponse += iResponseLen[jj];
   }
  }


  pResponse = gpResponses->index[0];

  memcpy( pResponse, szDeliveryResponse,
iResponseLen[DELIVERY_RESPONSE] );
  pResponse += iResponseLen[DELIVERY_RESPONSE];

  memcpy( pResponse, szNewOrderResponse,
iResponseLen[NEW_ORDER_RESPONSE] );
```

```
  pResponse += iResponseLen[NEW_ORDER_RESPONSE];

  memcpy(pResponse, szOrderStatusResponse,
iResponseLen[ORDER_STATUS_RESPONSE]);
  pResponse += iResponseLen[ORDER_STATUS_RESPONSE];

  memcpy( pResponse, szPaymentResponse,
iResponseLen[PAYMENT_RESPONSE] );
  pResponse += iResponseLen[PAYMENT_RESPONSE];

  memcpy( pResponse, szStockLevelResponse,
iResponseLen[STOCK_LEVEL_RESPONSE] );
  pResponse += iResponseLen[STOCK_LEVEL_RESPONSE];


  pResponse = gpResponses->index[0];
  for( ii = 1; ii < dwResponseSize; ii++ )
  {
    memcpy( gpResponses->index[ii], pResponse, iRowSize );
  }
}


void
DeleteTemplatePool( void )
{
  int                                   jj;

  for( jj = 0; jj < NUMBER_POOL_RESPONSE_TYPES; jj++ )
  {
    DeleteCriticalSection( &gpResponses->critSec[jj] );
  }
  free( gpResponses );

  for( jj = 0; jj < NUMBER_POOL_FORM_TYPES; jj++ )
  {
    DeleteCriticalSection( &gpForms->critSec[jj] );
  }
  free( gpForms );

  DeleteCriticalSection( &gpPanicForms->critSec );
  free( gpPanicForms );
}


void
MakeTransactionPool( DWORD dwTransactionPoolSize )
{
  int iMaxSize;
  int iSize;
  char *data;
  DWORD ii;


  iMaxSize = 0;
  iMaxSize = MAX(iMaxSize,sizeof(DeliveryData));
  iMaxSize = MAX(iMaxSize,sizeof(NewOrderData));
  iMaxSize = MAX(iMaxSize,sizeof(OrderStatusData));
  iMaxSize = MAX(iMaxSize,sizeof(PaymentData));
  iMaxSize = MAX(iMaxSize,sizeof(StockLevelData));
  iMaxSize = MAX(iMaxSize,sizeof(LoginData));
  iSize = (((char *)&gpTransactionPool->index - (char *)gpTransactionPool) +
                  (((char *)gpTransactionPool->data - (char *)gpTransactionPool-
>index)
                   * dwTransactionPoolSize ) +
                  (sizeof( char ) * iMaxSize * dwTransactionPoolSize ));
  gpTransactionPool = malloc( iSize );
  InitializeCriticalSection( &gpTransactionPool->critSec );
#ifdef FFE_DEBUG
  gpTransactionPool->iMaxIndex = dwTransactionPoolSize - 1;
  gpTransactionPool->iTransactionSize = iMaxSize;
  gpTransactionPool->iHistoryId = 0;
#endif
  gpTransactionPool->iNextFree = 0;



  data = ((char *)&gpTransactionPool->index[0] +
                  (((char *)&gpTransactionPool->data[0] -
                    (char *)&gpTransactionPool->index[0]) *
                   dwTransactionPoolSize ));

  for( ii = 0; ii < dwTransactionPoolSize; ii++ ) {
    gpTransactionPool->index[ii] = data;
    data += iMaxSize;
  }
}


void
DeleteTransactionPool( void )
{
  DeleteCriticalSection( &gpTransactionPool->critSec );
  free( gpTransactionPool );
}



void
BeginCmd( EXTENSION_CONTROL_BLOCK *pECB )
{
  SendWelcomeForm(pECB);
}



void
CheckpointCmd(EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id
)
{
  int                                   retcode;
  CheckpointData            checkpoint;

  checkpoint.w_id = w_id;
  checkpoint.ld_id = ld_id;
  checkpoint.pCC = pECB;

  retcode = TPCCCheckpoint( &checkpoint );
  if ( ERR_SUCCESS == retcode ) {
    SendMainMenuForm(pECB, w_id, ld_id,
                                  "Checkpoint issued (non-blocking), completed
(blocking).");
  }
  else {
    SendErrorResponse( pECB, retcode, ERR_TYPE_WEBDLL,
                                    NULL, w_id, ld_id, NULL );
  }
}



void
CheckpointStartupCmd(EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id )
{
  int                                   retcode;

  retcode = CKPTStartup( );
  if ( ERR_SUCCESS == retcode ) {
    SendMainMenuForm(pECB, w_id, ld_id, "Checkpoint Startup Succeded.");
  }
  else {
    SendErrorResponse( pECB, retcode, ERR_TYPE_WEBDLL,
                                    NULL, w_id, ld_id, NULL );
```

```
 }
}



void
ClearCmd(EXTENSION_CONTROL_BLOCK *pECB)
{
 if ( bLog )
 {
  TPCCCloseLog( );
  TPCCOpenLog( );
 }

 SendWelcomeForm(pECB);
}



void
ExitCmd( EXTENSION_CONTROL_BLOCK *pECB )
{
 TPCCDisconnect( pECB );

 SendWelcomeForm( pECB );
}



void
MenuCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id )
{
 SendMainMenuForm(pECB, w_id, ld_id, NULL);
}



void
SubmitCmd( EXTENSION_CONTROL_BLOCK *pECB, int *w_id, int *ld_id )
{
 int             iStatus;
 LoginData login;
 char            *ptr;

 if ( !GetCharKeyValuePtr( pECB->lpszQueryString, '4', &ptr ) ||
     ( 0 == ( *w_id = atoi( ptr ))) ||
     ( *w_id < 0 ))
 {
  SendErrorResponse( pECB, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL,
                                    NULL, *w_id, -1, NULL );
  goto SubmitError;
 }

 if ( !GetCharKeyValuePtr( pECB->lpszQueryString, '5', &ptr ) ||
     ( 0 == ( *ld_id = atoi( ptr ))) ||
     ( *ld_id > 10 ) ||
     ( *ld_id < 0 ))
 {
  SendErrorResponse( pECB, ERR_D_ID_INVALID, ERR_TYPE_WEBDLL,
                                    NULL, *w_id, *ld_id, NULL );
  goto SubmitError;
 }

 login.w_id = *w_id;
 login.ld_id = *ld_id;
 login.pCC = pECB;
 strcpy( login.szServer, gszServer );
 strcpy( login.szDatabase, gszDatabase );
 strcpy( login.szUser, gszUser );
 strcpy( login.szPassword, gszPassword );
 sprintf( login.szApplication, "TPCC" );
 iStatus = TPCCConnect( &login );
 if( ERR_DB_SUCCESS != iStatus )
 {
  SendErrorResponse( pECB, iStatus, ERR_TYPE_WEBDLL,
                                    NULL, *w_id, *ld_id, NULL );
  goto SubmitError;
 }

 SendMainMenuForm(pECB, *w_id, *ld_id, NULL);
 return;

SubmitError:
 return;

}



void
MemoryCheckCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
 _ASSERTE( _CrtCheckMemory( ));

 SendErrorResponse( pECB, ERR_SUCCESS, ERR_TYPE_WEBDLL, NULL,
                                    w_id, ld_id, NULL );
}


BOOL
GetKeyValuePtr( char *szIPtr, char *szKey, char **pszOPtr )
{
 char *szPtr1, *szPtr2;

 *pszOPtr = szIPtr;
 while (*szIPtr)
 {
  szPtr1 = szIPtr;
  szPtr2 = szKey;

  while ( *szPtr1 && *szPtr2 && 0 == ( *szPtr1 - *szPtr2 ))
   szPtr1++, szPtr2++;

  if ( '=' == *szPtr1 && '\0' == *szPtr2 )
  {
   *pszOPtr = ++szPtr1;
   return TRUE;
  }

  szIPtr++;
 }

 return FALSE;
}


BOOL
GetCharKeyValuePtr( char *szIPtr, char cKey, char **pszOPtr )
{
 BOOL       bGotStart;

 *pszOPtr = szIPtr;
 bGotStart = FALSE;
 while( *szIPtr )
 {
  if( cKey == *szIPtr && '=' == *++szIPtr )
  {
   *pszOPtr = ++szIPtr;
   return TRUE;
  }
  while( *szIPtr )
  {
   if( '&' == *szIPtr )
```

```
    {
                szIPtr++;
                break;
    }
    szIPtr++;
  }
 }

 return FALSE;
}


BOOL
GetNumeric(char *ptr, int *iValue)
{
 int c;
 int total;
 BOOL bGotSomething = FALSE;

 c = (int)(unsigned char)*ptr++;

 total = 0;

 while ((c >= '0') && (c <= '9'))
 {
  total = 10 * total + (c - '0');
  c = (int)(unsigned char)*ptr++;
  bGotSomething = TRUE;
 }
 if(('\0' == c) || ('&' == c) && bGotSomething)
 {
  *iValue = total;
  return (TRUE);
 }
 else
 {
  *iValue = 0;
  return(FALSE);
 }
}


BOOL
GetWDID(char *ptr, int *lw_id, int *ld_id, char **optr)
{
 int c;
 int pc;
 int total;
 BOOL bGotSomething = FALSE;

 *lw_id = 0;
 *ld_id = 0;
 total = 0;

 *optr = ptr;
 pc = (int)(unsigned char)*ptr++;
 if((pc < '0') || (pc > '9'))
  return FALSE;

 c = (int)(unsigned char)*ptr++;

 while ((c >= '0') && (c <= '9'))
 {
  total = 10 * total + (pc - '0');
  pc = c;
  c = (int)(unsigned char)*ptr++;
  bGotSomething = TRUE;
 }
 if(('\0' == c) || ('&' == c) && bGotSomething)
 {
  *lw_id = total;
  *ld_id = (pc - '0') + 1;
  *optr = ptr;
  return TRUE;
 }
 else
  return FALSE;
}



BOOL
GetKeyValueString(char *szIPtr, char *szKey,
                  char *szValue, int iSize)
{
 char *ptr;

 if( !GetKeyValuePtr( szIPtr, szKey, &ptr ))
  return FALSE;


 iSize--;

 while( '\0' != *ptr && '&' != *ptr && iSize)
 {
  *szValue++ = *ptr++;
  iSize--;
 }
 *szValue = 0;
 return TRUE;
}



#ifdef FFE_DEBUG

unsigned __stdcall
CheckMemory(void *param)
{
        while (TRUE)
        {
                _ASSERTE(_CrtCheckMemory());
                Sleep(1000);
        }

        return 0;
}

#endif
```

Appendix B

Sybase Device Init and Database Create Code and Segment Creation

```
disk init
  name = 'ordlne01',
  physname = '/tpcc_devs/ordlne01',
  vdevno = 1,
  size = 8320000
go
disk init
  name = 'ordlne02',
  physname = '/tpcc_devs/ordlne02',
  vdevno = 2,
  size = 8320000
go
disk init
  name = 'ordlne03',
  physname = '/tpcc_devs/ordlne03',
  vdevno = 3,
  size = 8320000
go
disk init
  name = 'ordlne04',
  physname = '/tpcc_devs/ordlne04',
  vdevno = 4,
  size = 8320000
go
disk init
  name = 'orders01',
  physname = '/tpcc_devs/orders01',
  vdevno = 5,
  size = 554496
go
disk init
  name = 'orders02',
  physname = '/tpcc_devs/orders02',
  vdevno = 6,
  size = 554496
go
disk init
  name = 'orders03',
  physname = '/tpcc_devs/orders03',
  vdevno = 7,
  size = 554496
go
disk init
  name = 'history01',
  physname = '/tpcc_devs/history01',
  vdevno = 8,
  size = 3328000
go
disk init
  name = 'cidx01',
  physname = '/tpcc_devs/cidx01',
  vdevno = 9,
  size = 1152000
go
disk init
  name = 'cidx02',
  physname = '/tpcc_devs/cidx02',
  vdevno = 10,
  size = 1152000
go
disk init
  name = 'stock01',
  physname = '/tpcc_devs/stock01',
  vdevno = 11,
  size = 1534976
go
disk init
  name = 'stock02',
  physname = '/tpcc_devs/stock02',
  vdevno = 12,
  size = 1534976
go
disk init
  name = 'stock03',
  physname = '/tpcc_devs/stock03',
  vdevno = 13,
  size = 1534976
go
disk init
  name = 'stock04',
  physname = '/tpcc_devs/stock04',
  vdevno = 14,
  size = 1534976
go
disk init
  name = 'stock05',
  physname = '/tpcc_devs/stock05',
  vdevno = 15,
  size = 1534976
go
disk init
  name = 'stock06',
  physname = '/tpcc_devs/stock06',
  vdevno = 16,
  size = 1534976
go
disk init
  name = 'stock07',
  physname = '/tpcc_devs/stock07',
  vdevno = 17,
  size = 1534976
go
disk init
  name = 'stock08',
  physname = '/tpcc_devs/stock08',
  vdevno = 18,
  size = 1534976
go
disk init
  name = 'stock09',
  physname = '/tpcc_devs/stock09',
  vdevno = 19,
  size = 1534976
go
disk init
  name = 'stock10',
  physname = '/tpcc_devs/stock10',
  vdevno = 20,
  size = 1534976
go
disk init
  name = 'stock11',
  physname = '/tpcc_devs/stock11',
  vdevno = 21,
  size = 1534976
go
disk init
  name = 'stock12',
  physname = '/tpcc_devs/stock12',
  vdevno = 22,
  size = 1534976
go
disk init
  name = 'stock13',
  physname = '/tpcc_devs/stock13',
  vdevno = 23,
  size = 1534976
go
disk init
  name = 'stock14',
  physname = '/tpcc_devs/stock14',
  vdevno = 24,
  size = 1534976
go
disk init
  name = 'stock15',
  physname = '/tpcc_devs/stock15',
  vdevno = 25,
  size = 1534976
```

```
go
disk init
  name = 'stock16',
  physname = '/tpcc_devs/stock16',
  vdevno = 26,
  size = 1534976
go
disk init
  name = 'stock17',
  physname = '/tpcc_devs/stock17',
  vdevno = 27,
  size = 1534976
go
disk init
  name = 'stock18',
  physname = '/tpcc_devs/stock18',
  vdevno = 28,
  size = 1534976
go
disk init
  name = 'stock19',
  physname = '/tpcc_devs/stock19',
  vdevno = 29,
  size = 1534976
go
disk init
  name = 'stock20',
  physname = '/tpcc_devs/stock20',
  vdevno = 30,
  size = 1534976
go
disk init
  name = 'stock21',
  physname = '/tpcc_devs/stock21',
  vdevno = 31,
  size = 1534976
go
disk init
  name = 'stock22',
  physname = '/tpcc_devs/stock22',
  vdevno = 32,
  size = 1534976
go
disk init
  name = 'stock23',
  physname = '/tpcc_devs/stock23',
  vdevno = 33,
  size = 1534976
go
disk init
  name = 'stock24',
  physname = '/tpcc_devs/stock24',
  vdevno = 34,
  size = 1534976
go
disk init
  name = 'stock25',
  physname = '/tpcc_devs/stock25',
  vdevno = 35,
  size = 1534976
go
disk init
  name = 'stock26',
  physname = '/tpcc_devs/stock26',
  vdevno = 36,
  size = 1534976
go
disk init
  name = 'stock27',
  physname = '/tpcc_devs/stock27',
  vdevno = 37,
  size = 1534976
go
disk init
  name = 'stock28',
  physname = '/tpcc_devs/stock28',
  vdevno = 38,
  size = 1534976
go
disk init
  name = 'stock29',
  physname = '/tpcc_devs/stock29',
  vdevno = 39,
  size = 1534976
go
disk init
  name = 'stock30',
  physname = '/tpcc_devs/stock30',
  vdevno = 40,
  size = 1534976
go
disk init
  name = 'cust01',
  physname = '/tpcc_devs/cust01',
  vdevno = 41,
  size = 1814528
go
disk init
  name = 'cust02',
  physname = '/tpcc_devs/cust02',
  vdevno = 42,
  size = 1814528
go
disk init
  name = 'cust03',
  physname = '/tpcc_devs/cust03',
  vdevno = 43,
  size = 1814528
go
disk init
  name = 'cust04',
  physname = '/tpcc_devs/cust04',
  vdevno = 44,
  size = 1814528
go
disk init
  name = 'cust05',
  physname = '/tpcc_devs/cust05',
  vdevno = 45,
  size = 1814528
go
disk init
  name = 'cust06',
  physname = '/tpcc_devs/cust06',
  vdevno = 46,
  size = 1814528
go
disk init
  name = 'cust07',
  physname = '/tpcc_devs/cust07',
  vdevno = 47,
  size = 1814528
go
disk init
  name = 'cust08',
  physname = '/tpcc_devs/cust08',
  vdevno = 48,
  size = 1814528
go
disk init
  name = 'cust09',
  physname = '/tpcc_devs/cust09',
  vdevno = 49,
  size = 1814528
go
disk init
  name = 'cust10',
  physname = '/tpcc_devs/cust10',
  vdevno = 50,
  size = 1814528
go
disk init
  name = 'cust11',
  physname = '/tpcc_devs/cust11',
  vdevno = 51,
  size = 1814528
go
disk init
```

```
  name = 'cust12',
  physname = '/tpcc_devs/cust12',
  vdevno = 52,
  size = 1814528
go
disk init
  name = 'cust13',
  physname = '/tpcc_devs/cust13',
  vdevno = 53,
  size = 1814528
go
disk init
  name = 'cust14',
  physname = '/tpcc_devs/cust14',
  vdevno = 54,
  size = 1814528
go
disk init
  name = 'cust15',
  physname = '/tpcc_devs/cust15',
  vdevno = 55,
  size = 1814528
go
disk init
  name = 'log01',
  physname = '/tpcc_devs/log01',
  vdevno = 56,
  size = 5120000
go
create database tpcc
on master = 1300, ordlne01 = 16250
, ordlne02 = 16250
, ordlne03 = 16250
, ordlne04 = 16250
, orders01 = 1083
, orders02 = 1083
, orders03 = 1083
, history01 = 6500
, cidx01 = 2200
, cidx02 = 2200
, stock01 = 2998
, stock02 = 2998
, stock03 = 2998
, stock04 = 2998
, stock05 = 2998
, stock06 = 2998
, stock07 = 2998
, stock08 = 2998
, stock09 = 2998
, stock10 = 2998
, stock11 = 2998
, stock12 = 2998
, stock13 = 2998
, stock14 = 2998
, stock15 = 2998
, stock16 = 2998
, stock17 = 2998
, stock18 = 2998
, stock19 = 2998
, stock20 = 2998
, stock21 = 2998
, stock22 = 2998
, stock23 = 2998
, stock24 = 2998
, stock25 = 2998
, stock26 = 2998
, stock27 = 2998
, stock28 = 2998
, stock29 = 2998
, stock30 = 2998
, cust01 = 3544
, cust02 = 3544
, cust03 = 3544
, cust04 = 3544
, cust05 = 3544
, cust06 = 3544
, cust07 = 3544
, cust08 = 3544
, cust09 = 3544
, cust10 = 3544
, cust11 = 3544
, cust12 = 3544
, cust13 = 3544
, cust14 = 3544
, cust15 = 3544

log on log01 = 10000
go
use tpcc
go
sp_addsegment Scache , tpcc , master
go
sp_addsegment Scidx , tpcc , cidx01
go
sp_extendsegment Scidx , tpcc ,  cidx02
go
sp_addsegment Scust , tpcc , cust01
go
sp_extendsegment Scust , tpcc ,  cust02
go
sp_extendsegment Scust , tpcc ,  cust03
go
sp_extendsegment Scust , tpcc ,  cust04
go
sp_extendsegment Scust , tpcc ,  cust05
go
sp_extendsegment Scust , tpcc ,  cust06
go
sp_extendsegment Scust , tpcc ,  cust07
go
sp_extendsegment Scust , tpcc ,  cust08
go
sp_extendsegment Scust , tpcc ,  cust09
go
sp_extendsegment Scust , tpcc ,  cust10
go
sp_extendsegment Scust , tpcc ,  cust11
go
sp_extendsegment Scust , tpcc ,  cust12
go
sp_extendsegment Scust , tpcc ,  cust13
go
sp_extendsegment Scust , tpcc ,  cust14
go
sp_extendsegment Scust , tpcc ,  cust15
go
sp_addsegment Shistory , tpcc , history01
go
sp_addsegment Sorders , tpcc , orders01
go
sp_extendsegment Sorders , tpcc ,  orders02
go
sp_extendsegment Sorders , tpcc ,  orders03
go
sp_addsegment Sordlne , tpcc , ordlne01
go
sp_extendsegment Sordlne , tpcc ,  ordlne02
go
sp_extendsegment Sordlne , tpcc ,  ordlne03
go
sp_extendsegment Sordlne , tpcc ,  ordlne04
go
sp_addsegment Sstock , tpcc , stock01
go
sp_extendsegment Sstock , tpcc ,  stock02
go
sp_extendsegment Sstock , tpcc ,  stock03
go
sp_extendsegment Sstock , tpcc ,  stock04
go
sp_extendsegment Sstock , tpcc ,  stock05
go
sp_extendsegment Sstock , tpcc ,  stock06
go
sp_extendsegment Sstock , tpcc ,  stock07
go
sp_extendsegment Sstock , tpcc ,  stock08
go
```

```
sp_extendsegment Sstock , tpcc ,  stock09
go
sp_extendsegment Sstock , tpcc ,  stock10
go
sp_extendsegment Sstock , tpcc ,  stock11
go
sp_extendsegment Sstock , tpcc ,  stock12
go
sp_extendsegment Sstock , tpcc ,  stock13
go
sp_extendsegment Sstock , tpcc ,  stock14
go
sp_extendsegment Sstock , tpcc ,  stock15
go
sp_extendsegment Sstock , tpcc ,  stock16
go
sp_extendsegment Sstock , tpcc ,  stock17
go
sp_extendsegment Sstock , tpcc ,  stock18
go
sp_extendsegment Sstock , tpcc ,  stock19
go
sp_extendsegment Sstock , tpcc ,  stock20
go
sp_extendsegment Sstock , tpcc ,  stock21
go
sp_extendsegment Sstock , tpcc ,  stock22
go
sp_extendsegment Sstock , tpcc ,  stock23
go
sp_extendsegment Sstock , tpcc ,  stock24
go
sp_extendsegment Sstock , tpcc ,  stock25
go
sp_extendsegment Sstock , tpcc ,  stock26
go
sp_extendsegment Sstock , tpcc ,  stock27
go
sp_extendsegment Sstock , tpcc ,  stock28
go
sp_extendsegment Sstock , tpcc ,  stock29
go
sp_extendsegment Sstock , tpcc ,  stock30
go
use tpcc
go
sp_dropsegment 'default', tpcc , cidx01
go
sp_dropsegment 'system', tpcc , cidx01
go
sp_dropsegment 'default', tpcc , cidx02
go
sp_dropsegment 'system', tpcc , cidx02
go
sp_dropsegment 'default', tpcc , cust01
go
sp_dropsegment 'system', tpcc , cust01
go
sp_dropsegment 'default', tpcc , cust02
go
sp_dropsegment 'system', tpcc , cust02
go
sp_dropsegment 'default', tpcc , cust03
go
sp_dropsegment 'system', tpcc , cust03
go
sp_dropsegment 'default', tpcc , cust04
go
sp_dropsegment 'system', tpcc , cust04
go
sp_dropsegment 'default', tpcc , cust05
go
sp_dropsegment 'system', tpcc , cust05
go
sp_dropsegment 'default', tpcc , cust06
go
sp_dropsegment 'system', tpcc , cust06
go
sp_dropsegment 'default', tpcc , cust07
go
sp_dropsegment 'system', tpcc , cust07
go
sp_dropsegment 'default', tpcc , cust08
go
sp_dropsegment 'system', tpcc , cust08
go
sp_dropsegment 'default', tpcc , cust09
go
sp_dropsegment 'system', tpcc , cust09
go
sp_dropsegment 'default', tpcc , cust10
go
sp_dropsegment 'system', tpcc , cust10
go
sp_dropsegment 'default', tpcc , cust11
go
sp_dropsegment 'system', tpcc , cust11
go
sp_dropsegment 'default', tpcc , cust12
go
sp_dropsegment 'system', tpcc , cust12
go
sp_dropsegment 'default', tpcc , cust13
go
sp_dropsegment 'system', tpcc , cust13
go
sp_dropsegment 'default', tpcc , cust14
go
sp_dropsegment 'system', tpcc , cust14
go
sp_dropsegment 'default', tpcc , cust15
go
sp_dropsegment 'system', tpcc , cust15
go
sp_dropsegment 'default', tpcc , history01
go
sp_dropsegment 'system', tpcc , history01
go
sp_dropsegment 'default', tpcc , orders01
go
sp_dropsegment 'system', tpcc , orders01
go
sp_dropsegment 'default', tpcc , orders02
go
sp_dropsegment 'system', tpcc , orders02
go
sp_dropsegment 'default', tpcc , orders03
go
sp_dropsegment 'system', tpcc , orders03
go
sp_dropsegment 'default', tpcc , ordlne01
go
sp_dropsegment 'system', tpcc , ordlne01
go
sp_dropsegment 'default', tpcc , ordlne02
go
sp_dropsegment 'system', tpcc , ordlne02
go
sp_dropsegment 'default', tpcc , ordlne03
go
sp_dropsegment 'system', tpcc , ordlne03
go
sp_dropsegment 'default', tpcc , ordlne04
go
sp_dropsegment 'system', tpcc , ordlne04
go
sp_dropsegment 'default', tpcc , stock01
go
sp_dropsegment 'system', tpcc , stock01
go
sp_dropsegment 'default', tpcc , stock02
go
sp_dropsegment 'system', tpcc , stock02
go
sp_dropsegment 'default', tpcc , stock03
go
sp_dropsegment 'system', tpcc , stock03
go
```

```
sp_dropsegment 'default', tpcc , stock04
go
sp_dropsegment 'system', tpcc , stock04
go
sp_dropsegment 'default', tpcc , stock05
go
sp_dropsegment 'system', tpcc , stock05
go
sp_dropsegment 'default', tpcc , stock06
go
sp_dropsegment 'system', tpcc , stock06
go
sp_dropsegment 'default', tpcc , stock07
go
sp_dropsegment 'system', tpcc , stock07
go
sp_dropsegment 'default', tpcc , stock08
go
sp_dropsegment 'system', tpcc , stock08
go
sp_dropsegment 'default', tpcc , stock09
go
sp_dropsegment 'system', tpcc , stock09
go
sp_dropsegment 'default', tpcc , stock10
go
sp_dropsegment 'system', tpcc , stock10
go
sp_dropsegment 'default', tpcc , stock11
go
sp_dropsegment 'system', tpcc , stock11
go
sp_dropsegment 'default', tpcc , stock12
go
sp_dropsegment 'system', tpcc , stock12
go
sp_dropsegment 'default', tpcc , stock13
go
sp_dropsegment 'system', tpcc , stock13
go
sp_dropsegment 'default', tpcc , stock14
go
sp_dropsegment 'system', tpcc , stock14
go
sp_dropsegment 'default', tpcc , stock15
go
sp_dropsegment 'system', tpcc , stock15
go
sp_dropsegment 'default', tpcc , stock16
go
sp_dropsegment 'system', tpcc , stock16
go
sp_dropsegment 'default', tpcc , stock17
go
sp_dropsegment 'system', tpcc , stock17
go
sp_dropsegment 'default', tpcc , stock18
go
sp_dropsegment 'system', tpcc , stock18
go
sp_dropsegment 'default', tpcc , stock19
go
sp_dropsegment 'system', tpcc , stock19
go
sp_dropsegment 'default', tpcc , stock20
go
sp_dropsegment 'system', tpcc , stock20
go
sp_dropsegment 'default', tpcc , stock21
go
sp_dropsegment 'system', tpcc , stock21
go
sp_dropsegment 'default', tpcc , stock22
go
sp_dropsegment 'system', tpcc , stock22
go
sp_dropsegment 'default', tpcc , stock23
go
sp_dropsegment 'system', tpcc , stock23
go
sp_dropsegment 'default', tpcc , stock24
go
sp_dropsegment 'system', tpcc , stock24
go
sp_dropsegment 'default', tpcc , stock25
go
sp_dropsegment 'system', tpcc , stock25
go
sp_dropsegment 'default', tpcc , stock26
go
sp_dropsegment 'system', tpcc , stock26
go
sp_dropsegment 'default', tpcc , stock27
go
sp_dropsegment 'system', tpcc , stock27
go
sp_dropsegment 'default', tpcc , stock28
go
sp_dropsegment 'system', tpcc , stock28
go
sp_dropsegment 'default', tpcc , stock29
go
sp_dropsegment 'system', tpcc , stock29
go
use master
go
checkpoint
go
```

Sybase Table and Index Definition

```
#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF

/* This script will create all the tables required for TPC-C benchmark */
/* It will also create some of the indexes. */
sp_dboption tpcc,"select into/bulkcopy",true
go
use tpcc
go
checkpoint
go

if exists ( select name from sysobjects where name = 'warehouse')
        drop table warehouse
go
create table warehouse (
        w_id                    smallint,
        w_name                  char(10),
        w_street_1      char(20),
        w_street_2      char(20),
        w_city                  char(20),
        w_state                 char(2),
        w_zip                   char(9),
        w_tax                   real,
        w_ytd                   float                   /*- Updated by
PID, PNM */
) on Scache
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                   real,
        d_ytd                   float,                  /*- Updated by
PID, PNM */
        d_next_o_id int                 /*- Updated by NO */
) on Scache
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  real,
        c_delivery_cnt          smallint,
        c_payment_cnt           smallint,   /*- Updated by PNM, PID */
        c_balance   float,                  /*- Updated by PNM, PID */
        c_ytd_payment           float,                  /*- Updated by
PNM, PID */
        c_data1                 char(250),  /*- Updated (?) by PNM, PID
*/
        c_data2                 char(250)   /*- Updated (?) by PNM, PID
*/
) on Scustomer
go
create unique clustered index c_clu
        on customer(c_w_id, c_id, c_d_id)
        on Scustomer
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    float,
        h_data                  char(24)
) on Shistory
go
/* alter table history unpartition */
alter table history partition 8
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 Scache
go
create unique clustered index no_clu
        on new_order(no_w_id, no_d_id, no_o_id)
        on Scache
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go

if exists ( select name from sysobjects where name = 'orders')
        drop table orders
go
create table orders (
        o_id                    int,
        o_c_id                  int,
        o_d_id                  tinyint,
        o_w_id                  smallint,
        o_entry_d   datetime,
        o_carrier_id smallint,  /*- Updated by D */
        o_ol_cnt    tinyint,
        o_all_local tinyint
) on Sorders
go

create unique clustered index o_clu
        on orders(o_w_id, o_d_id, o_id)
        on Sorders
go


dbcc tune(ascinserts, 1, orders)
go
dbcc tune(oamtrips, 100, orders)
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,   /*- Updated by D */
        ol_quantity smallint,
        ol_amount   float,
        ol_dist_info char(24)
) on Sorder_line
go
create unique clustered index ol_clu
        on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
        on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
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                 float,
        i_data                  char(50)
) on Scache
go
create unique clustered index i_clu
        on item(i_id)
        on Scache
go
dbcc tune(indextrips, 10, item)
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,   /*- Updated by NO */
        s_ytd                   int,                    /*- Updated by
NO */
```

```
        s_order_cnt  smallint,      /*- Updated by NO */
        s_remote_cnt                smallint,       /*- Updated by NO */
        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_data                      char(50)
) on Sstock
go
create unique clustered index s_clu
        on stock(s_i_id, s_w_id)
        on Sstock
go
dbcc tune(indextrips, 10, stock)
go

checkpoint
go
EOF



#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF
/* This script will create the TPC-C indexes that are best
  created after the load. */
use tpcc
go

create unique clustered index w_clu
        on warehouse(w_id)
        with fillfactor = 1
        on Scache
go
dbcc tune(indextrips, 100, warehouse)
go

create unique clustered index d_clu
        on district(d_w_id, d_id)
        with fillfactor = 1
        on Scache
go
dbcc tune(indextrips, 100, district)
go

create unique nonclustered index c_non1
        on customer(c_w_id, c_d_id, c_last, c_first, c_id)
        on Scidx
go

checkpoint
go
EOF



Sybase Data Load


typedef     unsigned long           BitVector;
#define WSZ                         (sizeof(BitVector)*8)
/* For axposf use WAREBATCH 0f 150 */
#define     WAREBATCH               150
#define     nthbit(map,n)           map[(n)/WSZ] &  (((BitVector)0x1)<<
((n)%WSZ))
#define     setbit(map,n)           map[(n)/WSZ] |= (((BitVector)0x1)<<
((n)%WSZ))

/*************************************************************
Load TPCC tables
*************************************************************/
```

```
#include "stdio.h"
#include "string.h"
#include "loader.h"

int load_item;
int load_warehouse;
int load_district;
int load_history;
int load_orders;
int load_new_order;
int load_order_line;
int load_customer;
int load_stock;
ID  w1, w2;
ID  warehouse;
int batch_size = 1000;
char   password[10];

int main(argn, argv)
        int argn;
        char **argv;
{

        /* Setup to use  the dblib version 10 for numeric datatypes */
    dbsetversion(DBVERSION_100);


        getargs(argn, argv);
        Randomize();

        if (load_item)                  LoadItems();
        if (load_warehouse)     LoadWarehouse(w1, w2);
        if (load_district)      LoadDistrict(w1, w2);
        if (load_history)       LoadHist(w1, w2);
        if (load_customer)      LoadCustomer(w1, w2);
        if (load_stock)                 LoadStock(w1, w2);
        if (load_orders)        LoadOrd(w1, w2);
        if (load_new_order)     LoadNew(w1, w2);
        return 0;
}



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

Warehouse

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


ID   w_id;
TEXT w_name[10+1];
TEXT w_street_1[20+1];
TEXT w_street_2[20+1];
TEXT w_city[20+1];
TEXT w_state[2+1];
TEXT w_zip[9+1];
FLOAT w_tax;
MONEY w_ytd;


int bulk_w;

LoadWarehouse(w1, w2)
        ID w1, w2;
{

        begin_warehouse_load();
        for (warehouse=w1; warehouse<=w2; warehouse++)
        {
                printf("Loading warehouse for warehouse %d\n",
warehouse);
```

```
                        w_id = warehouse;
                        MakeAlphaString(6, 10, w_name);
                        MakeAddress(w_street_1, w_street_2, w_city, w_state,
w_zip);

                        w_tax = RandomNumber(10, 20) / 100.0;
                        w_ytd = 300000.00;

                        warehouse_load();

                        printf("loaded warehouse for warehouse %d\n",
warehouse);
                }
                end_warehouse_load();
                return;
}


begin_warehouse_load()
{
                int             i = 1;

                bulk_w = bulk_open("tpcc", "warehouse", password);

                bulk_bind(bulk_w, i++, "w_id",   &w_id, ID_T);
                bulk_bind(bulk_w, i++, "w_name", w_name, TEXT_T);
                bulk_bind(bulk_w, i++, "w_street_1", w_street_1, TEXT_T);
                bulk_bind(bulk_w, i++, "w_street_2", w_street_2, TEXT_T);
                bulk_bind(bulk_w, i++, "w_city", w_city, TEXT_T);
                bulk_bind(bulk_w, i++, "w_state", w_state, TEXT_T);
                bulk_bind(bulk_w, i++, "w_zip", w_zip, TEXT_T);
                bulk_bind(bulk_w, i++, "w_tax", &w_tax, FLOAT_T);
                bulk_bind(bulk_w, i++, "w_ytd", &w_ytd, MONEY_T);
}


warehouse_load()
{
                debug("Loading Warehouse %d\n", w_id);
                bulk_load(bulk_w);
}


end_warehouse_load()
{
                bulk_close(bulk_w);
}




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

District

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



ID d_id;
ID d_w_id;
TEXT d_name[10+1];
TEXT d_street_1[20+1];
TEXT d_street_2[20+1];
TEXT d_city[20+1];
TEXT d_state[2+1];
TEXT d_zip[9+1];
FLOAT d_tax;
MONEY d_ytd;
ID   d_next_o_id;

int bulk_d;

LoadDistrict(w1, w2)
                ID w1, w2;
{
                ID w_id;

                begin_district_load();
                for (w_id=w1; w_id<=w2; w_id++)
                {
                        printf("Loading districts for warehouse %d\n", w_id);

                        d_w_id = w_id;
                        d_ytd = 30000.00;
                        d_next_o_id = 3001;

                        for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
                        {

                                MakeAlphaString(6, 10, d_name);
                                MakeAddress(d_street_1, d_street_2,
d_city, d_state, d_zip);
                                d_tax = RandomNumber(10,20) / 100.0;

                                district_load();
                        }
                        printf("loaded district for warehouse %d\n", w_id);
                }
                end_district_load();
                return;
}



begin_district_load()
{
                int             i = 1;

                bulk_d = bulk_open("tpcc", "district", password);

                bulk_bind(bulk_d, i++, "d_id", &d_id, ID_T);
                bulk_bind(bulk_d, i++, "d_w_id", &d_w_id, ID_T);
                bulk_bind(bulk_d, i++, "d_name", d_name, TEXT_T);
                bulk_bind(bulk_d, i++, "d_street_1", d_street_1, TEXT_T);
                bulk_bind(bulk_d, i++, "d_street_2", d_street_2, TEXT_T);
                bulk_bind(bulk_d, i++, "d_city", d_city, TEXT_T);
                bulk_bind(bulk_d, i++, "d_state", d_state, TEXT_T);
                bulk_bind(bulk_d, i++, "d_zip", d_zip, TEXT_T);
                bulk_bind(bulk_d, i++, "d_tax", &d_tax, FLOAT_T);
                bulk_bind(bulk_d, i++, "d_ytd", &d_ytd, MONEY_T);
                bulk_bind(bulk_d, i++, "d_next_o_id", &d_next_o_id, ID_T);
}

district_load()
{
                debug("District %d  w_id=%d\n", d_id, d_w_id);
                bulk_load(bulk_d);
}

end_district_load()
{
                bulk_close(bulk_d);
}



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

Item

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

```
ID i_id;
ID i_im_id;
TEXT i_name[24+1];
MONEY i_price;
TEXT  i_data[50+1];

int bulk_i;

LoadItems()
{
        int perm[MAXITEMS+1];
        int i, r, t;

        printf("Loading items\n");

        begin_item_load();

        /* select exactly 10% of items to be labeled "original" */
        RandomPermutation(perm, MAXITEMS);

        /* do for each item */
        for (i_id=1;   i_id <= MAXITEMS; i_id++)
        {

        /* Generate Item Data */
        MakeAlphaString(14, 24, i_name);
        i_price = RandomNumber(100,10000) / 100.0;
        MakeAlphaString(26, 50, i_data);
        if (perm[i_id] <= (MAXITEMS+9)/10)
                Original(i_data);

        /* Generate i_im_id for V 3.0 */
        i_im_id = RandomNumber(1, 10000);

         item_load();
         }

        end_item_load();
        return;
}



begin_item_load()
{
        int             i = 1;

        bulk_i = bulk_open("tpcc", "item", password);

        /* bind the variables to the sybase columns */
        bulk_bind(bulk_i, i++, "i_id", &i_id, ID_T);
        bulk_bind(bulk_i, i++, "i_im_id", &i_im_id, ID_T);
        bulk_bind(bulk_i, i++, "i_name", i_name, TEXT_T);
        bulk_bind(bulk_i, i++, "i_price", &i_price, MONEY_T);
        bulk_bind(bulk_i, i++, "i_data", i_data, TEXT_T);
}



item_load()
{
        debug("i_id=%3d price=%5.2f data=%s\n",
          i_id,         i_price,        i_data);
        bulk_load(bulk_i);
}


end_item_load()
{
        bulk_close(bulk_i);
}
```

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

History

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



ID h_c_id;
ID h_c_d_id;
ID h_c_w_id;
ID h_d_id;
ID h_w_id;
DATE h_date;
MONEY h_amount;
TEXT h_data[24+1];

int bulk_h;

LoadHist(w1, w2)
        ID w1, w2;
{
        ID w_id;
        ID d_id, c_id;

        begin_history_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {

                for (d_id=1;  d_id <= DIST_PER_WARE; d_id++)
                {
                        for (c_id=1; c_id <= CUST_PER_DIST;
c_id++)
                                LoadCustHist(w_id, d_id,
c_id);
                }

                printf("\nLoaded history for warehouse %d\n", w_id);
        }
        end_history_load();
}


LoadCustHist(w_id, d_id, c_id)
        ID w_id, d_id, c_id;
{

        h_c_id = c_id;
        h_c_d_id = d_id;
        h_c_w_id = w_id;
        h_d_id = d_id;
        h_w_id = w_id;
        h_amount = 10.0;
        MakeAlphaString(12, 24,h_data);
        datetime(&h_date);
        history_load();
}



begin_history_load()
{
        int             i = 1;

        bulk_h = bulk_open("tpcc", "history", password);

        bulk_bind(bulk_h, i++, "h_c_id",   &h_c_id,   ID_T);
        bulk_bind(bulk_h, i++, "h_c_d_id", &h_c_d_id, ID_T);
        bulk_bind(bulk_h, i++, "h_c_w_id", &h_c_w_id, ID_T);
        bulk_bind(bulk_h, i++, "h_d_id",   &h_d_id,   ID_T);
        bulk_bind(bulk_h, i++, "h_w_id",   &h_w_id,   ID_T);
```

```
        bulk_bind(bulk_h, i++, "h_date",   &h_date,   DATE_T);
        bulk_bind(bulk_h, i++, "h_amount", &h_amount, MONEY_T);
        bulk_bind(bulk_h, i++, "h_data",   h_data,    TEXT_T);
}

history_load()
{

        debug("h_c_id=%d  h_amount=%g\n", h_c_id, h_amount);
        bulk_load(bulk_h);
}


end_history_load()
{
        bulk_close(bulk_h);
}



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

Customer

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

/* static variables containing fields for customer record */
ID c_id;
ID c_d_id;
ID c_w_id;
TEXT c_first[16+1];
TEXT c_middle[2+1] = "OE";
TEXT c_last[16+1];
TEXT c_street_1[20+1];
TEXT c_street_2[20+1];
TEXT c_city[20+1];
TEXT c_state[2+1];
TEXT c_zip[9+1];
TEXT c_phone[16+1];
DATE c_since;
TEXT c_credit[2+1] = "?C";
MONEY c_credit_lim = 50000.0;       /* is this money or long or float? */
FLOAT c_discount;
MONEY c_balance = -10.0;
MONEY c_ytd_payment = 10.0;
COUNT c_payment_cnt = 1;
COUNT c_delivery_cnt = 0;
TEXT c_data[500+1];
TEXT c_data1[250+1];
TEXT c_data2[250+1];
ID          len;

int bulk_c;


LoadCustomer(w1, w2)
        ID w1, w2;
{
        ID w_id;

        begin_customer_load();
        for (w_id=w1; w_id<=w2; w_id++)
        {
                Customer(w_id);
                printf("\nLoaded customer for warehouse %d\n", w_id);
        }
        end_customer_load();
}

Customer(w_id)
        int w_id;
{
        BitVector badcredit[DIST_PER_WARE][(3000+WSZ-1)/WSZ], *
bmp;
        int i, j;
        ID d_id;

        /* Mark exactly 10% of customers as having bad credit */
        for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
        {
                bmp = badcredit[d_id-1];
                for (i=0; i<(3000+WSZ-1)/WSZ; i++)
                        bmp[i] = (BitVector)0x0000;
                for (i=0; i<(3000+9)/10; i++)
                {
                        do {
                                j = RandomNumber(0,3000-1);
                        } while (nthbit(bmp,j));
                        setbit(bmp,j);
                }
        }

        c_w_id = w_id;
        for (i=0; i<CUST_PER_DIST; i++)
        {
                c_id = i+1;
                for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
                {
                        c_d_id = d_id;

        LastName(i<1000?i:NURandomNumber(255,123,0,999),c_last);
                        MakeAlphaString(8, 16, c_first);

        MakeAddress(c_street_1,c_street_2,c_city,c_state,c_zip);
                        MakeNumberString(16, 16, c_phone);
                        MakeAlphaString(300, 500, c_data);
                        datetime(&c_since);
                        c_credit[0] = nthbit(badcredit[d_id-1],i) ? 'B'
: 'G';
                        c_discount = RandomNumber(0, 50) / 100.0;
                        c_balance = -10.0;
                        customer_load();
                }
        }
}

begin_customer_load()
{
        int         i = 1;

        bulk_c = bulk_open("tpcc", "customer", password);

        bulk_bind(bulk_c, i++, "c_id",          &c_id,   ID_T);
        bulk_bind(bulk_c, i++, "c_d_id",        &c_d_id, ID_T);
        bulk_bind(bulk_c, i++, "c_w_id",        &c_w_id, ID_T);
        bulk_bind(bulk_c, i++, "c_first",   c_first, TEXT_T);
        bulk_bind(bulk_c, i++, "c_middle",  c_middle, TEXT_T);
        bulk_bind(bulk_c, i++, "c_last",        c_last,   TEXT_T);
        bulk_bind(bulk_c, i++, "street_1",  c_street_1 , TEXT_T);
        bulk_bind(bulk_c, i++, "street_2",  c_street_2,  TEXT_T);
        bulk_bind(bulk_c, i++, "c_city",        c_city,          TEXT_T);
        bulk_bind(bulk_c, i++, "c_state",   c_state,         TEXT_T);
        bulk_bind(bulk_c, i++, "c_zip",         c_zip,           TEXT_T);
        bulk_bind(bulk_c, i++, "c_phone",   c_phone,         TEXT_T);
        bulk_bind(bulk_c, i++, "c_since",   &c_since,        DATE_T);
        bulk_bind(bulk_c, i++, "c_credit",  c_credit,        TEXT_T);
        bulk_bind(bulk_c, i++, "c_credit_lim", &c_credit_lim, MONEY_T);
        bulk_bind(bulk_c, i++, "c_discount", &c_discount, FLOAT_T);
        bulk_bind(bulk_c, i++, "c_delivery_cnt", &c_delivery_cnt,
COUNT_T);
        bulk_bind(bulk_c, i++, "c_payment_cnt",&c_payment_cnt,
COUNT_T);
        bulk_bind(bulk_c, i++, "c_balance", &c_balance, MONEY_T);
        bulk_bind(bulk_c, i++, "c_ytd_payment", &c_ytd_payment,
MONEY_T);
        bulk_bind(bulk_c, i++, "c_data_1", c_data1, TEXT_T);
        bulk_bind(bulk_c, i++, "c_data_2", c_data2, TEXT_T);
}

customer_load()
{
```

```
		debug("c_id=%-5d  d_id=%-5d  w_id=%-5d  c_last=%s\n",
		  c_id,		c_d_id,  c_w_id,  c_last);


		/* Break the string c_data into 2 pieces */
		len = strlen(c_data);
		if (len > 250)
			{
			memcpy(c_data1, c_data, 250);
			c_data1[250]='\0';
			memcpy(c_data2, c_data+250, len-250 +1);
			}

		else
			{
			memcpy(c_data1, c_data, 250+1);
			strcpy(c_data2,"");
			}

		/* load the data */
		bulk_load(bulk_c);
}

end_customer_load()
{
		bulk_close(bulk_c);
}



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

Order, Order line, New order

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

/* Order row */
ID o_id;
ID o_c_id;
ID o_d_id;
ID o_w_id;
DATE o_entry_d;
ID o_carrier_id;
COUNT o_ol_cnt;
LOGICAL o_all_local;

/* Order line row */
ID   ol_o_id;
ID   ol_d_id;
ID   ol_w_id;
ID   ol_number;
ID   ol_i_id;
ID   ol_supply_w_id;
DATE ol_delivery_d;
COUNT ol_quantity;
MONEY ol_amount;
TEXT  ol_dist_info[24+1];

/* new order row */
ID no_o_id;
ID no_d_id;
ID no_w_id;


int  o_bulk;
int ol_bulk;
int no_bulk;



LoadOrd(w1, w2)
		ID w1, w2;
{
		ID w_id;
		ID d_id;

		begin_order_load();
		begin_order_line_load();
		for (w_id=w1; w_id<=w2; w_id++)
		{

			for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
				Orders(w_id, d_id);

			printf("\nLoaded order + order_line for warehouse %d\n",
w_id);
		}
		end_order_line_load();
		end_order_load();
}

LoadNew(w1, w2)
		ID w1, w2;
{
		ID w_id;
		ID d_id;

		begin_new_order_load();
		for (w_id=w1; w_id<=w2; w_id++)
		{
			for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
			{
				no_d_id = d_id;
				no_w_id = w_id;
				for (no_o_id=2101;  no_o_id <=
ORD_PER_DIST; no_o_id++)
					new_order_load();
			}
			printf("\nLoaded new_order for warehouse %d\n", w_id);
		}
		end_new_order_load();
}

Orders(w_id, d_id)
		ID w_id;
		ID d_id;
{
		int cust[ORD_PER_DIST+1];
		int ol_cnt[ORD_PER_DIST+1], sum;
		ID ol;

		printf("\nLoading orders and order lines  for warehouse %d district
%d\n",
		 w_id, d_id);

		RandomPermutation(cust, ORD_PER_DIST);


		for (o_id = 1, sum=0;  o_id <= ORD_PER_DIST;  o_id++)
	    sum += (ol_cnt[o_id] = RandomNumber(5, 15));

	while (sum > 10*ORD_PER_DIST)
	{
	    do {
		o_id = RandomNumber(1,ORD_PER_DIST);
	    } while (ol_cnt[o_id]==5);
	    ol_cnt[o_id]--;
	    sum--;
	}

	while (sum < 10*ORD_PER_DIST)
	{
	    do {
		o_id = RandomNumber(1,ORD_PER_DIST);
	    } while (ol_cnt[o_id]==15);
	    ol_cnt[o_id]++;
	    sum++;
	}

		for (o_id = 1;  o_id <= ORD_PER_DIST;  o_id++)
		{
		o_c_id = cust[o_id];
```

```
		o_d_id = d_id;
		o_w_id = w_id;
		datetime(&o_entry_d);
		if (o_id <= 2100)
		o_carrier_id = RandomNumber(1,10);
		else o_carrier_id = -1;
		o_ol_cnt = ol_cnt[o_id];
		/* o_ol_cnt = RandomNumber(5, 15); */
		o_all_local = 1;
		order_load();

		for (ol=1;  ol<=o_ol_cnt; ol++)
			OrderLine(ol);

		}
}



OrderLine(ol)
		ID ol;
{

		ol_o_id = o_id;
		ol_d_id = o_d_id;
		ol_w_id = o_w_id;
		ol_number = ol;
		ol_i_id = RandomNumber(1, MAXITEMS);
		ol_supply_w_id = o_w_id;
		ol_delivery_d = o_entry_d;
		ol_quantity = 5;
		if (o_id <= 2100) ol_amount = 0;
		else 			ol_amount = RandomNumber(1, 999999) /
100.0;
		MakeAlphaString(24, 24, ol_dist_info);
		order_line_load();
}


NewOrder(w_id, d_id)
		ID w_id, d_id;
{

		no_d_id = o_d_id;
		no_w_id = o_w_id;
		for (no_o_id=2101;  no_o_id <= ORD_PER_DIST; no_o_id++)
		new_order_load();
}



begin_order_load()
{
		int		i = 1;

		o_bulk = bulk_open("tpcc", "orders", password);

		bulk_bind(o_bulk, i++, "o_id", &o_id, ID_T);
		bulk_bind(o_bulk, i++, "o_c_id", &o_c_id, ID_T);
		bulk_bind(o_bulk, i++, "o_d_id", &o_d_id, ID_T);
		bulk_bind(o_bulk, i++, "o_w_id", &o_w_id, ID_T);
		bulk_bind(o_bulk, i++, "o_entry_d", &o_entry_d, DATE_T);
		bulk_bind(o_bulk, i++, "o_carrier_id", &o_carrier_id, ID_T);
		bulk_bind(o_ol_cnt, i++, "o_ol_cnt", &o_ol_cnt, COUNT_T);
		bulk_bind(o_all_local, i++, "o_all_local", &o_all_local,
LOGICAL_T);
}

order_load()
{
		debug("o_id=%d o_c_id=%d  count=%d\n", o_id, o_c_id, o_ol_cnt);
		bulk_load(o_bulk);
}

end_order_load()
{
		bulk_close(o_bulk);
}



begin_order_line_load()
{
		int		i = 1;

		ol_bulk = bulk_open("tpcc", "order_line", password);

		bulk_bind(ol_bulk, i++, "ol_o_id", &ol_o_id, ID_T);
		bulk_bind(ol_bulk, i++, "ol_d_id", &ol_d_id, ID_T);
		bulk_bind(ol_bulk, i++, "ol_w_id", &ol_w_id, ID_T);
		bulk_bind(ol_bulk, i++, "ol_number", &ol_number, ID_T);
		bulk_bind(ol_bulk, i++, "ol_i_id", &ol_i_id, ID_T);
		bulk_bind(ol_bulk, i++, "ol_supply_w_id", &ol_supply_w_id, ID_T);
		bulk_bind(ol_bulk, i++, "ol_delivery_d", &ol_delivery_d, DATE_T);
		bulk_bind(ol_bulk, i++, "ol_quantity", &ol_quantity, COUNT_T);
		bulk_bind(ol_bulk, i++, "ol_amount", &ol_amount, MONEY_T);
		bulk_bind(ol_bulk, i++, "ol_dist_info", ol_dist_info, TEXT_T);
}

order_line_load()
{
		static int ol_count = 0;
		debug("  ol_o_id=%d  ol_number=%d  ol_amount=%g\n",
			ol_o_id,	ol_number,	ol_amount);
		bulk_load(ol_bulk);
}


end_order_line_load()
{
		bulk_close(ol_bulk);
}

begin_new_order_load()
{
		int		i = 1;

		no_bulk = bulk_open("tpcc", "new_order", password);

		bulk_bind(no_bulk, i++, "no_o_id", &no_o_id, ID_T);
		bulk_bind(no_bulk, i++, "no_d_id", &no_d_id, ID_T);
		bulk_bind(no_bulk, i++, "no_w_id", &no_w_id, ID_T);
}

new_order_load()
{
		debug("  no_o_id=%d  \n", no_o_id);
		bulk_load(no_bulk);
}


end_new_order_load()
{
		bulk_close(no_bulk);
}


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

Stock

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


ID s_i_id;
ID s_w_id;
COUNT s_quantity;
TEXT s_dist_01[24+1];
TEXT s_dist_02[24+1];
```

```
TEXT s_dist_03[24+1];
TEXT s_dist_04[24+1];
TEXT s_dist_05[24+1];
TEXT s_dist_06[24+1];
TEXT s_dist_07[24+1];
TEXT s_dist_08[24+1];
TEXT s_dist_09[24+1];
TEXT s_dist_10[24+1];
COUNT s_ytd;
COUNT s_order_cnt;
COUNT s_remote_cnt;
TEXT s_data[50+1];

int bulk_s;

/*
** On loading stock in major order of item_id:
** 10% of the MAXITEMS items in each warehouse need to marked as original
** (i.e., s_data like '%ORIGINAL%'.)  This is a bit harder to do when we
** load by item number, rather than by warehouses.  The trick is to first
** generate a huge WAREBATCH * MAXITEMS bitmap, initialize all bits to zero,
** and then set 10% of bits in each row to 1.  While loading item i in
** warehouse w, we simply lookup bitmap[w][i] to see whether it needs to
** be marked as original.
*/

LoadStock(w1, w2)
        ID w1, w2;
{
        ID w_id;
        BitVector original[WAREBATCH][((MAXITEMS+(WSZ-1))/WSZ)],
* bmp;
        int w, i, j;


        if (w2-w1+1 > WAREBATCH)
        {
                fprintf(stderr, "Can't load stock for %d warehouses.\n",
                        w2-w1+1);
                fprintf(stderr, "Please use batches of %d.\n",
WAREBATCH);
        }

        for (w=w1; w<=w2; w++)
        {
                bmp = original[w-w1];
                /* Mark all items as not "original" */
                for (i=0; i<(MAXITEMS+(WSZ-1))/WSZ; i++)
                        bmp[i] = (BitVector)0x0000;
                /* Mark exactly 10% of items as "original" */
                for (i=0; i<(MAXITEMS+9)/10; i++)
                {
                        do {
                                j =
RandomNumber(0,MAXITEMS-1);
                        } while (nthbit(bmp,j));
                        setbit(bmp,j);
                }
        }

        printf("Loading stock for warehouse %d to %d.\n", w1, w2);
        begin_stock_load();
        /* do for each item */
        for (s_i_id=1;  s_i_id <= MAXITEMS; s_i_id++)
        {
                for (w_id=w1; w_id<=w2; w_id++)
                {
                /* Generate Stock Data */
                        s_w_id = w_id;
                        s_quantity = RandomNumber(10,100);
                        MakeAlphaString(24, 24, s_dist_01);
                        MakeAlphaString(24, 24, s_dist_02);
                        MakeAlphaString(24, 24, s_dist_03);
                        MakeAlphaString(24, 24, s_dist_04);
                        MakeAlphaString(24, 24, s_dist_05);
                        MakeAlphaString(24, 24, s_dist_06);
                        MakeAlphaString(24, 24, s_dist_07);
                        MakeAlphaString(24, 24, s_dist_08);
                        MakeAlphaString(24, 24, s_dist_09);
                        MakeAlphaString(24, 24, s_dist_10);
                        s_ytd = 0;
                        s_order_cnt = 0;
                        s_remote_cnt = 0;
                        MakeAlphaString(26, 50, s_data);
                        if (nthbit(original[w_id-w1],s_i_id-1))
                        {
                                Original(s_data);
                        }
                        stock_load();
                }
        }
        end_stock_load();
        printf("\nLoaded stock for warehouses %d to %d.\n", w1, w2);
}

begin_stock_load()
{
        int             i = 1;

        bulk_s = bulk_open("tpcc", "stock", password);

        bulk_bind(bulk_s, i++, "s_i_id", &s_i_id, ID_T);
        bulk_bind(bulk_s, i++, "s_w_id", &s_w_id, ID_T);
        bulk_bind(bulk_s, i++, "s_quantity", &s_quantity, COUNT_T);
        bulk_bind(bulk_s, i++, "s_ytd",          &s_ytd,      COUNT_T);
        bulk_bind(bulk_s, i++, "s_order_cnt", &s_order_cnt, COUNT_T);
        bulk_bind(bulk_s, i++, "s_remote_cnt", &s_remote_cnt, COUNT_T);
        bulk_bind(bulk_s, i++, "s_dist_01", s_dist_01, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_02", s_dist_02, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_03", s_dist_03, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_04", s_dist_04, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_05", s_dist_05, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_06", s_dist_06, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_07", s_dist_07, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_08", s_dist_08, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_09", s_dist_09, TEXT_T);
        bulk_bind(bulk_s, i++, "s_dist_10", s_dist_10, TEXT_T);
        bulk_bind(bulk_s, i++, "s_data",         s_data,         TEXT_T);
}



stock_load()
{
        debug("s_i_id=%d w_id=%d s_data=%s\n",
          s_i_id, s_w_id,   s_data);
        bulk_load(bulk_s);
}


end_stock_load()
{
        bulk_close(bulk_s);
}



test(){}


getargs(argc, argv)

/*************************************************************
**
configure configures the load stuff
  By default, loads all the tables for a the specified warehouse.
                        When loading warehouse 1, also loads the item table.
*************************************************************
****/
        int argc;
        char **argv;
{
        char ch;

        /* define the defaults */
        load_item = load_warehouse = load_district = load_history =
        load_orders = load_new_order = load_order_line =
```

```
        load_customer = load_stock = NO;


        if              (strcmp(argv[1], "warehouse") == 0) load_warehouse =
YES;
        else if (strcmp(argv[1], "district") == 0)  load_district = YES;
        else if (strcmp(argv[1], "stock") == 0)  load_stock = YES;
        else if (strcmp(argv[1], "item") == 0)  load_item = YES;
        else if (strcmp(argv[1], "history") == 0)  load_history = YES;
        else if (strcmp(argv[1], "orders") == 0)  load_orders = YES;
        else if (strcmp(argv[1], "customer") == 0)  load_customer = YES;
        else if (strcmp(argv[1], "new_order") ==0) load_new_order = YES;
        else
        {
                printf("%s is not a valid table name\n", argv[1]);
                exit(0);

        }

        /* Set the w1 and w2 to argv[2] and arg[3] */
        if (argc < 3)
        {
                printf("Usage: %s <table> <w_first> [<w_last>]\n",
argv[0]);
                exit(1);
        }
        {
                w1 = atoi(argv[2]);
                if (argc >= 3)
                        w2 = atoi(argv[3]);
                else
                        w2 = w1;
        }

        /* Get the password for sa */
        if (argc > 4)
        strcpy(password,argv[4]);

        /* Check if warehouse is within the range */
        if (w1 <= 0 || w2 > 1000 || w1 > w2)
        {
                printf("Warehouse id is out of range\n");
                exit(0);
        }
}

double drand48();

MakeAddress(str1, str2, city, state, zip)
        TEXT str1[20+1];
        TEXT str2[20+1];
        TEXT city[20+1];
        TEXT state[2+1];
        TEXT zip[9+1];
{
        MakeAlphaString(10,20,str1);
        MakeAlphaString(10,20,str2);
        MakeAlphaString(10,20,city);
        MakeAlphaString(2,2,state);
        MakezipString(0,9999,zip);

        /* Changed for TPCC V 3.0 */
        strcat(zip, "11111");

}



LastName(num, name)
/*****************************************************************
Lastname generates a lastname from a number.
*****************************************************************/
        int num;
        char name[20+1];
{
        int i;
        static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",

        "ESE", "ANTI", "CALLY", "ATION", "EING"};

        strcpy(name, n[(num/100)%10]);
        strcat(name, n[(num/10) %10]);
        strcat(name, n[(num/1)  %10]);
}


int MakeNumberString(min, max, num)
        int min;
        int max;
        TEXT num[];
{
        static char digit[]="0123456789";
        int length;
        int i;

        length = RandomNumber(min, max);

        for (i=0;  i<length;  i++)
        num[i] = digit[RandomNumber(0,9)];
        num[length] = '\0';

        return length;
}

int MakezipString(min, max, num)
      int min;
      int max;
      TEXT num[];
{
      static char digit[]="0123456789";
      int length;
      int i;

      length = 4;

      for (i=0;  i<length;  i++)
      num[i] = digit[RandomNumber(0,9)];
      num[length] = '\0';

      return length;
}



int MakeAlphaString(min, max, str)
        int min;
        int max;
        TEXT str[];
{
        static char character[] =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567
89";
        int length;
        int i;

        length = RandomNumber(min, max);

        for (i=0;  i<length;  i++)
        str[i] = character[RandomNumber(0, sizeof(character)-2)];
        str[length] = '\0';

        return length;
}


Original(str)
        TEXT str[];
{
        int pos;
        int len;

        len = strlen(str);
        if (len < 8) return;

        pos = RandomNumber(0,len-8);

        str[pos+0] = 'O';
        str[pos+1] = 'R';
```

```
        str[pos+2] = 'I';
        str[pos+3] = 'G';
        str[pos+4] = 'I';
        str[pos+5] = 'N';
        str[pos+6] = 'A';
        str[pos+7] = 'L';
}



RandomPermutation(perm, n)
        int perm[];
        int n;
{
        int i, r, t;

        /* generate the identity permutation to start with */
        for (i=1; i<=n; i++)
        perm[i] = i;

        /* randomly shuffle the permutation */
        for (i=1; i<=n; i++)
        {
        r = RandomNumber(i, n);
        t = perm[i]; perm[i] = perm[r]; perm[r] = t;
        }
}



int Randomize()
{
        srand48(time(0)+getpid());
}



int RandomNumber(min, max)
        int min;
        int max;
{
        int r;
        r = (int)(drand48() * (max - min + 1)) + min;
        return r;
}


int NURandomNumber(a, c, min, max)
        int a;
        int c;
        int min;
        int max;
{
        int r;

        r = ((RandomNumber(0, a) | RandomNumber(min, max)) + c)
        % (max - min + 1) + min;

        return r;
}
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED

#include <sybfront.h>
#include <sybdb.h>
#include <time.h>


/* Population constants */
#ifdef CACHED
#define MAXITEMS 10000
#define CUST_PER_DIST 300
#define DIST_PER_WARE 10
#define ORD_PER_DIST 300
#else
#define MAXITEMS 100000
#define CUST_PER_DIST 3000
#define DIST_PER_WARE 10
#define ORD_PER_DIST 3000
#endif

/* Types of application variables */
typedef int     COUNT;
typedef int     ID;
typedef double  MONEY;
typedef double  FLOAT;
typedef char    TEXT;
typedef struct { int x[2];} DATE;
typedef int     LOGICAL;

typedef enum
   {COUNT_T, ID_T, MONEY_T, FLOAT_T, TEXT_T,
    DATE_T, LOGICAL_T, MAX_T}
   DATA_TYPE;

typedef struct timeval TIME;




#define YES 1
#define NO 0
#define EOF (-1)

#ifndef NULL
#define NULL ((void *)0)
#endif

#ifdef DEBUG
#define debug printf
#else
#define debug (void)
#endif

/* define function types */
extern          int             msg_handler();
extern          int             err_handler();
extern          int             batch_size;

#endif /* TPCC_INCLUDED */
#if ! lint
static char *sddsId = "@(#)  error.c  1.1  4/30/91  19:47:32";
#endif /* ! lint */

/*
** Confidential property of Sybase, Inc.
** (c) Copyright Sybase, Inc. 1991
** All rights reserved
*/

/*
** error.c:     1.1     4/30/91 19:47:32
**      Standard error handler for RungenII and supporting code
**
**      HMS [04/30/91]
*/

/* Required standard include files */
#include <stdio.h>
#ifdef _NTINTEL
#include <stdlib.h>
#include <windows.h>
#endif

/* Required Sybase include files */
#include <sybfront.h>
#include <sybdb.h>

/* message numbers that we don't want to deal with */
#define DUMB_MESSAGE    5701
#define ABORT_ERROR     6104


#ifdef _NTINTEL
int
 err_handler(dbproc, severity, errno, oserr, errstr, oserrstr)
```

```
  DBPROCESS *dbproc;
  int severity;
  int errno;
  int oserr;
  char  *errstr;
  char  *oserrstr;
{

      /* changing databases message */
      if (errno == DUMB_MESSAGE || errno == ABORT_ERROR)
            return(INT_CANCEL);

      fprintf(stderr,"DB-LIBRARY Error: \n\t%s\n",errstr);

      if (oserr != DBNOERR)
            fprintf(stderr,"O/S Error: \n\t%s\n",oserrstr);

      /* exit on any error */
      exit(-100);
}
#else
int
err_handler(dbproc, severity, errno, oserr)
  DBPROCESS *dbproc;
  int severity;
  int errno;
  int oserr;
{
      /* changing databases message */
      if (errno == DUMB_MESSAGE || errno == ABORT_ERROR)
            return(INT_CANCEL);

      fprintf(stderr,"DB-LIBRARY Error: \n\t%s\n",dberrstr(errno));

      if (oserr != DBNOERR)
            fprintf(stderr,"O/S Error: \n\t%s\n",dboserrstr(oserr));

      /* exit on any error */
      exit(-100);
}
#endif


int
msg_handler(dbproc,msgno,msgstate,severity,msgtext,servername,procname,line)
DBPROCESS      *dbproc;
int            msgno;
int            msgstate;
int            severity;
char           *msgtext;
char           *servername;
char           *procname;
int            line;
{

      /* changing database messages */
      if (msgno == DUMB_MESSAGE || msgno == ABORT_ERROR || msgno ==
5703 || msgno == 5704 || msgno == 4843)
            return(SUCCEED);

      /* Is this a deadlock message */
      if (msgno == 1205)
      {
            /* Set the deadlock indicator */
            *((DBBOOL *) dbgetuserdata(dbproc)) = TRUE;

            /* Sleep a few seconds before going back */
#ifdef _NTINTEL
            Sleep((DWORD) 2000);
#else
            sleep((unsigned) 2);
#endif
            return(SUCCEED);

      }

      fprintf(stderr, "msg no %d -\n%s", msgno, msgtext);

      /* exit on any error */
      exit(-101);
}

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

Sybase Specific Routines

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

#include <stdio.h>
#include <sys/time.h>
#include <string.h>
#include "loader.h"

datetime(date)
            DBDATETIME *date;
{
            struct timeval time;
            gettimeofday(&time, NULL);
            date->dtdays = time.tv_sec / (60*60*24)
                        + (1970-1900)*365 + (1970-1900)/4;
            date->dttime = (time.tv_sec % (60*60*24))*300
                        + time.tv_usec*300/1000000;
}


/* define the type information for each field */
typedef struct
{
            char *terminator;
            int termlen;
            int type;
} bind_parm;

bind_parm parm[MAX_T] =
{
            /* COUNT */          {NULL, 0, SYBINT4},
            /* ID    */          {NULL, 0, SYBINT4},
            /* MONEY */          {NULL, 0, SYBFLT8},
            /* FLOAT */          {NULL, 0, SYBFLT8},
            /* TEXT */  {"",   1, SYBCHAR},
            /* DATE */  {NULL, 0, SYBDATETIME},
            /* LOGICAL */        {NULL, 0, SYBINT4}
};

#define MAXOPENS 10

DBPROCESS *dbproc[MAXOPENS];
int         count[MAXOPENS];

int bulk_open(database, table, password)
            char database[];
            char table[];
            char password[];
{
            LOGINREC *login;
            int db;

            /* make note we have established a connection */
            for (db=0; db<MAXOPENS; db++)
                        if (dbproc[db] == NULL) break;
            count[db] = 0;

            /* Install an error and Message handler */
            dbmsghandle(msg_handler);
            dberrhandle(err_handler);

            /* initialize dblib */
            if (dbinit() != SUCCEED)
                        printf("Can't initialize the DB library\n");

            /* allocate a login record and fill it in */
            login = dblogin();
```

```
        if (login == NULL)
                printf("Can't allocate a login record.\n");
        DBSETLUSER(login, "sa");

        if(strlen(password) > 0)
        DBSETLPWD(login, password);

        DBSETLAPP(login, table);
        BCP_SETL(login, TRUE);

        /* Set Packet Size to 4096 */
        DBSETLPACKET(login, 4096);

        /* establish a connection with the server specified by DSQUERY */
        dbproc[db] = dbopen(login, NULL);
        if (dbproc[db] == NULL)
                printf("Can't establish connection. Is DSQUERY set?\n");

        /* select the database to use */
        if (database != NULL)
                if (dbuse(dbproc[db], database) != SUCCEED)
                        printf("Can't select database: %s\n",
database);

        /* release the login record */
        dbloginfree(login);

        /* prepare to do a bulk copy */
        if (bcp_init(dbproc[db], table, NULL, NULL, DB_IN) != SUCCEED)
        printf("Can't initialize the bulk copy to table %s\n", table);

        return db;
}



bulk_bind(db, column, name, address, type)
        int db;
        int column;
        char name[];
        void *address;
        int type;
{
        if (bcp_bind(dbproc[db], address, 0, -1, parm[type].terminator,
                        parm[type].termlen, parm[type].type,
column) != SUCCEED)
                printf("Can't bind column %d to 0x%x, type=%d\n",
                        column,address,type);
}

bulk_null(db, column)
        int db;
        int column;
{
        if (bcp_collen(dbproc[db], 0, column) != SUCCEED)
                printf("Can't null column %d\n", column);
}


bulk_non_null(db, column)
        int db;
        int column;
{
        if (bcp_collen(dbproc[db], -1, column) != SUCCEED)
                printf("Can't non-null column %d\n", column);
}


bulk_load(db)
        int db;
{
        count[db]++;
        if (bcp_sendrow(dbproc[db]) != SUCCEED)
                printf("bulk_load: Can't load row\n");
        if (count[db]%batch_size == 0 && (bcp_batch(dbproc[db]) == -1))
        printf("bulk_load: Can't post rows\n");
        if (count[db]%1000 == 0) write(1,".",1);
        if (count[db]%50000 == 0) write(1,"\n",1);
}

bulk_close(db)
        int db;
{
        if (bcp_done(dbproc[db]) == -1)
                printf("Problems completing the bulk copy.\n");
        dbproc[db] = NULL;
        if (count[db] >= 1000) write(1,"\n",1);
}
```

tpcc_cache_bind.sh

```
#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF

use master
go
sp_dboption tpcc, "single user", true
go

use tpcc
go
checkpoint
go

/*
** Cache c_log
*/

sp_bindcache "c_log", "tpcc", "syslogs"
go

/*
** Cache c_tinyhot
*/

sp_bindcache "c_sysindex", "tpcc", "sysindexes"
go
sp_bindcache "c_sysindex", "tpcc", "sysindexes","sysindexes"
go

use master
go
sp_dboption tpcc, "single user", false
go

use tpcc
go
checkpoint
go

/*
** Cache c_tinyhot (continued)
*/

sp_bindcache "c_tinyhot", "tpcc", "item"
go
sp_bindcache "c_tinyhot", "tpcc", "item","i_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "warehouse"
go
sp_bindcache "c_tinyhot", "tpcc", "warehouse","w_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "district"
go
sp_bindcache "c_tinyhot", "tpcc", "district","d_clu"
go

/*
** Cache c_no_ol
*/

sp_bindcache "c_no_ol", "tpcc", "new_order"
```

```
go
sp_bindcache "c_no_ol", "tpcc", "new_order","no_clu"
go
sp_bindcache "c_no_ol", "tpcc", "order_line"
go

/*
** Cache c_ol_index
*/

sp_bindcache "c_ol_index", "tpcc", "order_line","ol_clu"
go

/*
** Cache c_orders
*/

sp_bindcache "c_orders", "tpcc", "orders"
go
sp_bindcache "c_orders", "tpcc", "orders","o_clu"
go

/*
** Cache c_stock_index
*/

sp_bindcache "c_stock_index", "tpcc", "stock","s_clu"
go

/*
** Cache c_stock
*/

sp_bindcache "c_stock", "tpcc", "stock"
go

/*
** Cache c_customer
*/

sp_bindcache "c_customer", "tpcc", "customer"
go
sp_bindcache "c_customer_index", "tpcc", "customer","c_clu"
go
sp_bindcache "c_customer_index", "tpcc", "customer","c_non1"
go
```

Appendix C

Remote Terminal Emulator

scr_util.c

```
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#include <stdarg.h>

#ifdef _WIN32
# include <nt_lib.h>
#endif


#include <prte.h>
#include <stdtypes.h>
#define SCR_UTIL_C
#include <scr_util.h>



static print_getnet_debug(char *name, char *ret_string, char *def_string);


int
SCR_find_reducer() {
  char *temp_ptr;

  temp_ptr = PRTEget_network_variable("REDUCER");
  if (temp_ptr != NULL)
    return atoi(temp_ptr);
  else return 0;
}



void
SCR_error(int errnum, char *wstring) {
  char *errorstrings[] = {
    "OKAY",
    "ERROR sending string:",
    "ERROR waiting after:",
    "ERROR send_and_sync sending:",
    "ERROR connecting:"
  };

  PRTEto_log(999,errorstrings[errnum]);
  PRTEto_log(999,wstring);
}

int32
SCRerr( int32 ErrCode )
{
  char                                  ErrFmt[] = "\007ERROR( %d ): %s\n\n";
  int32                                 Idx;

  Idx = 0;
  while( '\0' != ErrMsgs[Idx].ErrMsg[0] && ErrCode != ErrMsgs[Idx].ErrCode )
  {
    Idx++;
  }

  if( '\0' == ErrMsgs[Idx].ErrMsg[0] )
  {

    Idx = ERR_NO_ERR_MSG;
  }

  SCRlog( ErrFmt, ErrCode, ErrMsgs[Idx].ErrMsg );
  return( ErrCode );
}


int
SCR_to_sut_and_log ( char *message, char *sync_string) {
  PRTEto_log(666, message);
  PRTEsync_string(sync_string);
  PRTEsend(message);

  return PRTEwait_for();
}


void
SCR_to_console_and_log ( char *message) {
  PRTEto_log(666, message);
  PRTEsend_console_message(message);
}

void
SCRlog( char *Format, ... )
{
  char                          Buf[128];
  va_list       VarArgs;



  if( NULL != Format ) {
    va_start( VarArgs, Format );
    vsprintf( Buf, Format, VarArgs );
    va_end( VarArgs );
  }
  else
  {
    strcpy( Buf, "[NULL]" );
  }

  PRTEsend_console_message( Buf );
  PRTEto_log( 555, Buf );

  return;
}




void
SCR_inverse_random(double Mean, double Max, double *Iexp) {
  double Randval, lval;

  do {
    Randval = drand48();
    lval = log(Randval);
    *Iexp  = -1.0 * Mean * lval;

  } while (*Iexp > Max );
}

char *
SCR_getnet_int ( char *varname, char *default_str, int *pInt) {
  char *pnet;
```

```
  if (!varname) return NULL;

  pnet = PRTEget_network_variable (varname);
  if (pnet) *pInt = atoi(pnet);
  else *pInt = atoi(default_str);
#ifdef DEBUG
  print_getnet_debug(varname,pnet,default_str);
#endif

  return pnet;
}

char *
SCR_getnet_long ( char *varname, char *default_str, long *pLong) {
  char *pnet;

  if (!varname) return NULL;

  pnet = PRTEget_network_variable (varname);
  if (pnet) *pLong = atol(pnet);
  else *pLong = atol(default_str);
#ifdef DEBUG
  print_getnet_debug(varname,pnet,default_str);
#endif

  return pnet;
}

char *
SCR_getnet_float ( char *varname, char *default_str, float *pFloat) {
  char *pnet;

  if (!varname) return NULL;

  pnet = PRTEget_network_variable (varname);
  if (pnet) *pFloat = (float) atof(pnet);
  else *pFloat = (float) atof(default_str);
#ifdef DEBUG
  print_getnet_debug(varname,pnet,default_str);
#endif

  return pnet;
}

char *
SCR_getnet_double ( char *varname, char *default_str, double *pDouble) {
  char *pnet;

  if (!varname) return NULL;

  pnet = PRTEget_network_variable (varname);
  if (pnet) *pDouble = atof(pnet);
  else *pDouble = atof(default_str);

  return pnet;
}


char *
SCR_getnet_string ( char *varname, char *default_str, char *pString) {
  char *pnet;

  if (!varname) return NULL;

  pnet = PRTEget_network_variable (varname);

  if (pnet != NULL) strcpy(pString,pnet);

  else {
    if (default_str != NULL) strcpy(pString,default_str);
    else (pString[0] = '\0');
  }
#ifdef DEBUG
  print_getnet_debug(varname,pnet,default_str);
#endif

  return pString;
}

char *
SCR_getnet_bool ( char *varname, char *default_str, BOOL *pBool) {
  char *pnet;
  char first_letter;

  if (!varname) return NULL;

  pnet = PRTEget_network_variable (varname);
  if (pnet)
  {
    first_letter = pnet[0];
  }
  else
  {
    first_letter = default_str[0];
    pnet = default_str;
  }
  switch (first_letter)
  {
    case 't':
    case 'T':
    case '1':
      *pBool = TRUE;
      break;

    case 'f':
    case 'F':
    case '0':
    default:
      *pBool = FALSE;
      break;
    }
#ifdef DEBUG
  print_getnet_debug(varname,pnet,default_str);
#endif

  return pnet;
}

#ifdef DEBUG
static
print_getnet_debug(char *name, char *ret_string, char *def_string) {
  char temp_string[256];

  sprintf(temp_string,"Netvar: name = %s netvar = %s, default = %s\n",
                name,ret_string,def_string);
  SCR_to_console_and_log (temp_string);
}
#endif


char
*SCR_next_list_element (char *List, char *Element) {

  char *pTmpChar;

  pTmpChar = Element;

  if ( *List ) {
    while ( *List && *List != ',' && !isspace(*List) && *List != '"') {
      *pTmpChar++ = *List++;
    }
    *pTmpChar = '\0';
```

```
   while ( isspace(*List) || *List == ',' || *List == '"')
     List++;

   return List;
  }

  else return NULL;
}

int
SCR_count_elements (char *List) {

  char *pTmpChar;
  int Count = 1;


  pTmpChar = List;
  while (*pTmpChar++) {
    if (',' == *pTmpChar)
      Count++;
  }
  return Count;
}



void
SCR_webpage_callback ( void *Context, char *Data, int Bytes)
{
  web_page_t *WebPage = Context;

  memcpy( WebPage->pBuf, Data, Bytes );
  WebPage->pBuf += Bytes;
  *WebPage->pBuf = '\0';
}
```

scr_util.h

```
#ifndef SCR_UTIL_H
#define SCR_UTIL_H


#define BOOL          int
#define TRUE          1
#define FALSE         0



#define E_FAILURE -1
#define E_OKAY 0
#define E_SEND 1
#define E_WAIT 2
#define E_SENDSYNC 3
#define E_CONNECT 4



union dataptr {
  double *pDouble;
  int *pInt;
  float *pFloat;
};



#define MAX_WEB_PAGE_SIZE 3072
#define WEB_PAGE_SUCCESS 0
typedef struct _web_page_t
{
  BOOL Inited;
  int  HttpStatusOffset;
  int  FormStrOffset;
  int  FormStrLen;
  int  FormValOffset;
  int  ReqMinLen;
  char Buf[MAX_WEB_PAGE_SIZE];
  char *pBuf;
} web_page_t;

void SCR_webpage_callback (void *WebPage, char *Data, int Bytes);


void SCR_error(int errnum, char *string);
int SCR_to_sut_and_log (char *message, char *sync_string);
void SCR_to_console_and_log (char *message);
void SCR_inverse_random (double Mean, double Max, double *Iexp);

char *SCR_getnet_int     (char *varname, char *default_str, int *pInt);
char *SCR_getnet_float   (char *varname, char *default_str, float *pFloat);
char *SCR_getnet_long    (char *varname, char *default_str, long *pLong);
char *SCR_getnet_double (char *varname, char *default_str, double *pDouble);
char *SCR_getnet_bool    (char *varname, char *default_str, BOOL *pBool);
char *SCR_getnet_string (char *varname, char *default_str, char *pString);
void          SCRlog( char *Format, ... );
int32         SCRerr( int32 ErrCode );
char *SCR_next_list_element (char *List, char *Element);
int SCR_count_elements (char *List);



#define       ERR_SUCCESS
              0
#define ERR_NO_ERR_MSG
              1
#define ERR_INIT                                          2
#define ERR_MAKING_RUN_DIR                                3
#define ERR_NO_REDUCER
              4
#define ERR_CKPT_INT_TOO_LONG
              5
#define ERR_CKPT_INT_GREATER_THAN_MEASUREMENT_INT         6
#define ERR_MEASUREMENT_NOT_INTEGRAL_MULTIPLE
              7
#define ERR_WARMUP_TOO_SHORT
              8
#define ERR_ADMIN_FE_NAMES                                9
#define ERR_ADMIN_USER_FE_COUNT_MISMATCH                  10
#define ERR_MALLOC_ADMIN_USER_ARRAY
              11
#define ERR_MALLOC_ADMIN_FE_ARRAY                         12
#define ERR_GET_BE_INFO
              13
#define ERR_INVALID_LOOPBACK_MODE                         14
#define ERR_AUDIT_FUNC_NETVARS
              15
#define ERR_BE_NAMES
              16
#define ERR_MALLOC_BE_ARRAY
              17
#define ERR_CODE_VERSION_NOT_SET                          18
#define ERR_RUN_DIR_NOT_SET                               19
#define ERR_CODE_VERSION_MISMATCH                         20
#define ERR_OPEN_BIN_LOG_FILE
              21
#define ERR_WRITE_BIN_LOG_FILE_HEADER_SIZE                22
#define ERR_WRITE_BIN_LOG_FILE_HEADER
              23
```

```
#define ERR_MASTER_ID_NOT_SET
        24
#define ERR_RUN_NUMBER_NOT_SET
        25
#define ERR_VERSION_NUMBER_NOT_SET                  26
#define ERR_REDUCER_UPDATE_INTERVAL_NOT_SET         27
#define ERR_REDUCER_HEADER_INTERVAL_NOT_SET         28
#define ERR_OPEN_AIDE_DATA_FILE
        29
#define ERR_DATABASE_TYPE_NOT_SET                   30
#define ERR_INVALID_DATABASE_TYPE                   31
#define ERR_AIDE_INIT_TIMEOUT
        32
#define ERR_FE_USER_COUNTS_NOT_SET                  33
#define ERR_FE_NAMES_NOT_SET
        34
#define ERR_FE_AND_USER_COUNT_MISMATCH
        35
#define ERR_CALC_VALID_TXN_MIX
        36
#define ERR_REDUCER_INIT_TIMEOUT                    37
#define ERR_LOAD_USER_DLL_TIMEOUT                   38
#define ERR_CONNECT_APPLICATION
        39
#define ERR_SEND                                    40
#define ERR_WAIT_FOR
        41
#define ERR_HTTP_STATUS_OFFSET
        42
#define ERR_FORM_TYPE_OFFSET
        43
#define ERR_WEB_PAGE_TOO_SHORT
        44
#define ERR_WEB_PAGE_STATUS_CODE                    45
#define ERR_ERROR_STATUS_CODE
        46
#define ERR_ERROR_WEB_PAGE                          47
#define ERR_WRONG_WEB_PAGE                          48
#define ERR_AIDE_TASK_TIMEOUT
        49
#define ERR_AIDES_FAILED_TASK
        50
#define ERR_BE_CKPT_TIMEOUT                         51
#define ERR_BE_CKPT_FAILED                          52
#define ERR_AIDE_CKPT_INIT_TIMEOUT                  53
#define ERR_USER_LOGIN_TIMEOUT
        54
#define ERR_USER_LOGIN_FAILED
        55
#define ERR_LOG_DIR_NOT_SET                         56
#define ERR_CGI_SCRIPT_NAME_NOT_SET                 57
#define ERR_INVALID_TXN_TYPE_REQUESTED
        58
#define ERR_USER_RESUME_FAILED
        59
#define ERR_USER_RESUME_TIMEOUT
        60
#define ERR_USER_STOP_FAILED                        61
#define ERR_USER_STOP_TIMEOUT
        62
#define ERR_OPEN_USER_COUNT_FILE                    63
#define ERR_USER_COUNT_LOW_ON_FE                    64
#define ERR_USER_COUNT_HIGH_ON_FE                   65
#define ERR_USER_COUNTS_TIMEOUT
        66
#define ERR_AIDE_CKPT_RUN_TIMEOUT                   67
#define ERR_AIDE_CKPT_STOP_TIMEOUT                  68
#define ERR_WRITE_SENDERS_ID
        69
#define ERR_WRITE_MSG_TYPE                          70
#define ERR_WRITE_MSG_SIZE                          71
#define ERR_WRITE_MSG
        72
#define ERR_AIDE_EXIT_TIMEOUT
        73
#define ERR_OPEN_C_LAST_FILE                        74
#define ERR_SYNC_USER_TIMEOUT
        75
#define ERR_MALLOC_FE_ARRAY
        76
#define ERR_MALLOC_USER_STOP_ARRAY                  77
#define ERR_MALLOC_FE_NAME_ARRAY                    78
#define ERR_ACTION_NOT_SET                          79
#define ERR_MAX_W_ID_NOT_SET
        80
#define ERR_REDUCER_ID_NOT_SET
        81


typedef struct _err_msg_t
{
  int32                 ErrCode;
  char                  ErrMsg[256];
} err_msg_t;

#ifdef SCR_UTIL_C

err_msg_t ErrMsgs[] =
{
  { ERR_SUCCESS, "Success, no error." },
  { ERR_NO_ERR_MSG, "No error message available for this error code." },
  { ERR_INIT, "Failed to initialize." },
  { ERR_MAKING_RUN_DIR, "Failed to make the run directory." },
  { ERR_NO_REDUCER, "Failed to get the reducer's PRTE user id." },
  { ERR_CKPT_INT_TOO_LONG, "Checkpoint interval > 1800 seconds, this
violates the benchmark specification." },
  { ERR_CKPT_INT_GREATER_THAN_MEASUREMENT_INT, "Checkpoint
interval > measurement interval, this violates the benchmark specification." },
  { ERR_MEASUREMENT_NOT_INTEGRAL_MULTIPLE, "Measurement
interval is not an integral multiple of checkpoint interval, this violates the
benchmark specification." },
  { ERR_WARMUP_TOO_SHORT, "The warmup interval is too short to support
requested checkpoint interval." },
  { ERR_ADMIN_FE_NAMES, "Unable to get the list of admin FEs." },
  { ERR_ADMIN_USER_FE_COUNT_MISMATCH, "Count mismatch between
admin users and admin FEs." },
  { ERR_MALLOC_ADMIN_USER_ARRAY, "Unable to malloc space for
admin user array." },
  { ERR_MALLOC_ADMIN_FE_ARRAY, "Unable to malloc space for admin
FE array." },
  { ERR_GET_BE_INFO, "Problem getting back end information." },
  { ERR_INVALID_LOOPBACK_MODE, "Invalid loopback mode specified." },
  { ERR_AUDIT_FUNC_NETVARS, "Problem getting audit function network
variables." },
  { ERR_BE_NAMES, "Unable to get BE_NAMES network variable." },
  { ERR_MALLOC_BE_ARRAY, "Unable to malloc space for BE array." },
  { ERR_CODE_VERSION_NOT_SET, "Code version network variable not set."
},
  { ERR_RUN_DIR_NOT_SET, "Run directory network variable not set." },
  { ERR_CODE_VERSION_MISMATCH, "Code version mismatch with
Master." },
  { ERR_OPEN_BIN_LOG_FILE, "Failed to open the binary data log file." },
  { ERR_WRITE_BIN_LOG_FILE_HEADER_SIZE, "Error writing binary data
log file header size." },
  { ERR_WRITE_BIN_LOG_FILE_HEADER, "Error writing binary data log file
header." },
  { ERR_MASTER_ID_NOT_SET, "Master ID network variable not set." },
  { ERR_RUN_NUMBER_NOT_SET, "Run number network variable not set." },
  { ERR_VERSION_NUMBER_NOT_SET, "Version number network variable
not set." },
  { ERR_REDUCER_UPDATE_INTERVAL_NOT_SET, "Reducer update
interval network variable not set." },
```

```
  { ERR_REDUCER_HEADER_INTERVAL_NOT_SET, "Reducer header
update interval network variable not set." },
  { ERR_OPEN_AIDE_DATA_FILE, "Error opening aide data file." },
  { ERR_DATABASE_TYPE_NOT_SET, "Database type network variable not
set." },
  { ERR_INVALID_DATABASE_TYPE, "Invalid database type" },
  { ERR_AIDE_INIT_TIMEOUT, "Timed out waiting for Aides to initialize." },
  { ERR_FE_USER_COUNTS_NOT_SET, "FE user count network variable not
set." },
  { ERR_FE_NAMES_NOT_SET, "FE names network variable not set." },
  { ERR_FE_AND_USER_COUNT_MISMATCH, "The number of FEs listed
and the number of user counts listed don't match." },
  { ERR_CALC_VALID_TXN_MIX, "Failed to calculate a valid transaction
mix." },
  { ERR_REDUCER_INIT_TIMEOUT, "Timed out waiting for Reducer to
initialize." },
  { ERR_LOAD_USER_DLL_TIMEOUT, "Timed out waiting for user dlls to
load." },
  { ERR_CONNECT_APPLICATION, "Error trying to connect to the application
(SUT)." },
  { ERR_SEND, "Error sending data to the application (SUT)." },
  { ERR_WAIT_FOR, "Error waiting for reply from application (SUT)." },
  { ERR_HTTP_STATUS_OFFSET, "Couldn't determine HTTP status offset." },
  { ERR_FORM_TYPE_OFFSET, "Couldn't determine the form type offset." },
  { ERR_WEB_PAGE_TOO_SHORT, "Received a web page that is too short to
be from our dll." },
  { ERR_WEB_PAGE_STATUS_CODE, "Couldn't find form status code in web
page." },
  { ERR_ERROR_STATUS_CODE, "Received an error page from our dll, but
couldn't parse the status code." },
  { ERR_ERROR_WEB_PAGE, "Received and error page from our dll." },
  { ERR_WRONG_WEB_PAGE, "Received a valid web page from our dll, but it
isn't the one we expected." },
  { ERR_AIDE_TASK_TIMEOUT, "Timed out waiting for all aides to perform a
task." },
  { ERR_AIDES_FAILED_TASK, "One or more aides failed to perform a task."
},
  { ERR_BE_CKPT_TIMEOUT, "Timed out waiting for all aides to do their
backend checkpoint init." },
  { ERR_BE_CKPT_FAILED, "One or more aides failed to do their backend
checkpoint init." },
  { ERR_AIDE_CKPT_INIT_TIMEOUT, "Timed out waiting for all aides to do
their checkpoint init." },
  { ERR_USER_LOGIN_TIMEOUT, "Timed out waiting for all users to confirm
login." },
  { ERR_USER_LOGIN_FAILED, "A user failed to log in successfully." },
  { ERR_LOG_DIR_NOT_SET, "Log dir network variable not set." },
  { ERR_CGI_SCRIPT_NAME_NOT_SET, "CGI script name network variable
not set. " },
  { ERR_INVALID_TXN_TYPE_REQUESTED, "An invalid transcation type
was requested." },
  { ERR_USER_RESUME_FAILED, "User(s) failed to resume to start doing
transactions." },
  { ERR_USER_RESUME_TIMEOUT, "Timed out waiting for all users to
resume to start doing transactions." },
  { ERR_USER_STOP_FAILED, "User(s) failed to confirm stop." },
  { ERR_USER_RESUME_TIMEOUT, "Timed out waiting for all users to
confirm stop." },
  { ERR_OPEN_USER_COUNT_FILE, "Error doing fopen of user count file." },
  { ERR_USER_COUNT_LOW_ON_FE, "User count too low on FE." },
  { ERR_USER_COUNT_HIGH_ON_FE, "User count too high on FE.  Do you
have a browser window connected?" },
  { ERR_USER_COUNTS_TIMEOUT, "Timed out waiting to get user counts on
all FEs." },
  { ERR_AIDE_CKPT_RUN_TIMEOUT, "Timed out waiting for aides to start
checkpoint loop." },
  { ERR_AIDE_CKPT_STOP_TIMEOUT, "Timed out waiting for aides to stop
checkpoint loop." },
  { ERR_WRITE_SENDERS_ID, "Failed to write sender's id to binary log file."
},
  { ERR_WRITE_MSG_TYPE, "Failed to write message type to binary log file."
},
  { ERR_WRITE_MSG_SIZE, "Failed to write message size to binary log file." },
  { ERR_WRITE_MSG, "Failed to write message to binary log file." },
  { ERR_AIDE_EXIT_TIMEOUT, "Timed out waiting for all aides to exit." },
  { ERR_OPEN_C_LAST_FILE, "Unable to open file to write C_LAST
constant." },
  { ERR_SYNC_USER_TIMEOUT, "Timed out on delay to allow users to catch
up with in sync offset." },
  { ERR_MALLOC_FE_ARRAY, "Unable to malloc space for the fe array." },
  { ERR_MALLOC_USER_STOP_ARRAY, "Unable to malloc space for the
user stop array." },
  { ERR_MALLOC_FE_NAME_ARRAY, "Unable to malloc space to hold the
name of the FE." },
  { ERR_ACTION_NOT_SET, "Action network variable not set." },
  { ERR_MAX_W_ID_NOT_SET, "Maximum warehouse ID network variable
not set." },
  { ERR_REDUCER_ID_NOT_SET, "Reducer ID network variable not set." },
  { 0, "" }
};
#else
extern err_msg_t ErrMsgs[];
#endif

#endif
```

tpcc.c

```
/*+          FILE:          TPCC.C
 *                          Microsoft TPC-C Kit Ver. 3.00.000
 *                          Audited 08/23/96 By Francois Raab
 *
 *                          Copyright Microsoft, 1996
 *                          Copyright Compaq Computer Corp., 1997
 *
 *          PURPOSE:        Main module for TPCC.DLL which is an ISAPI service dll.
 *          Author:                 Philip Durr
 *                                                  philipdu@Microsoft.com
 *
 *          MODIFICATIONS:
 *
 *                          Routines substantially modified by:
 *                                  Anne Bradley    Compaq Computer Corp.
 *                                  Bill Carr       Compaq Computer Corp.
 *
 */
/*+*************************************************************************
 *                                      *
 * COPYRIGHT (c) 1997 BY                        *
 * COMPAQ COMPUTER CORPORATION, .               *
 * ALL RIGHTS RESERVED.                         *
 *                                      *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED  *
 * ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER  *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY  *
 * OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY  *
 * TRANSFERRED.                                 *
 *                                      *
 * THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE  *
 * AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY COMPAQ COMPUTER  *
 * CORPORATION.                                 *
 •                                      *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.               *
 *                                      *
 *                                      *
 *************************************************************************/


#include <windows.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>
```

```
#define TPCC_C

#include <tpccerr.h>
#include <tpccstruct.h>
#include <tpccapi.h>
#include <httpext.h>

#include <tpcc.h>
#include <web_ui.h>



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;
}



int ParseNewOrderQuery(char *pQueryString, NewOrderData
*pNewOrderData)
{
 char   *ptr;
 int                    i;
 short      items;
 BOOL       bCheck;
 char       *pProcessedQuery[MAXNEWORDERVALS];


 PARSE_QUERY_STRING(pQueryString, MAXNEWORDERVALS,
                        newOrderStrs, pProcessedQuery);

 if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
  return ERR_NEWORDER_FORM_MISSING_DID;

 GetNumeric(ptr, &pNewOrderData->d_id);
 if(0 == pNewOrderData->d_id)
  return ERR_NEWORDER_DISTRICT_INVALID;


 if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
  return ERR_NEWORDER_CUSTOMER_KEY;

 if( !GetNumeric(ptr, &pNewOrderData->c_id))
  return ERR_NEWORDER_CUSTOMER_INVALID;

 bCheck = FALSE;
 pNewOrderData->o_all_local = 1;


 for(i=0, items=0; i<15; i++)
 {
  if( !GetValuePtr(pProcessedQuery, i*3+IID00, &ptr))
   return ERR_NEWORDER_MISSING_IID_KEY;
  if(*ptr != '&' && *ptr)
  {
   if ( bCheck )
            return ERR_NEWORDER_ITEM_BLANK_LINES;
   if(!GetNumeric(ptr, &pNewOrderData->o_ol[i].ol_i_id))
            return ERR_NEWORDER_ITEMID_INVALID;

   if(!GetValuePtr(pProcessedQuery, i*3+SP00, &ptr))
            return ERR_NEWORDER_MISSING_SUPPW_KEY;
   if(!GetNumeric(ptr, &pNewOrderData->o_ol[i].ol_supply_w_id))
            return ERR_NEWORDER_SUPPW_INVALID;
   if ( pNewOrderData->o_all_local &&
             pNewOrderData->o_ol[i].ol_supply_w_id !=
             pNewOrderData->w_id )
            pNewOrderData->o_all_local = 0;
   if(!GetValuePtr(pProcessedQuery, i*3+QTY00, &ptr))
            return ERR_NEWORDER_MISSING_QTY_KEY;
   if(!GetNumeric(ptr, &pNewOrderData->o_ol[i].ol_quantity))
            return ERR_NEWORDER_QTY_INVALID;
   items++;
   if ( pNewOrderData->o_ol[i].ol_i_id >= 1000000 ||
             pNewOrderData->o_ol[i].ol_i_id < 1 )
            return ERR_NEWORDER_ITEMID_RANGE;
   if ( pNewOrderData->o_ol[i].ol_quantity >= 100 ||
             pNewOrderData->o_ol[i].ol_quantity < 1 )
            return ERR_NEWORDER_QTY_RANGE;
  }
  else
  {
   if(!GetValuePtr(pProcessedQuery, i*3+SP00, &ptr))
            return ERR_NEWORDER_MISSING_QTY_KEY;
   if(*ptr != '&' && *ptr)
            return ERR_NEWORDER_SUPPW_WITHOUT_ITEMID;

   if(!GetValuePtr(pProcessedQuery, i*3+QTY00, &ptr))
            return ERR_NEWORDER_MISSING_QTY_KEY;
   if(*ptr != '&' && *ptr)
            return ERR_NEWORDER_QTY_WITHOUT_ITEMID;
   bCheck = TRUE;
  }
 }
 if ( items == 0 )
  return ERR_NEWORDER_NOITEMS_ENTERED;

 pNewOrderData->o_ol_cnt = items;

 return ERR_SUCCESS;
}


int ParseOrderStatusQuery(char *pQueryString,
                                OrderStatusData *pOrderStatusData)
{
 char       szTmp[26];
 char   *ptr;
 char       *pSzTmp;
 char       *pProcessedQuery[MAXORDERSTATUSVALS];


 PARSE_QUERY_STRING(pQueryString, MAXORDERSTATUSVALS,
                        orderStatusStrs, pProcessedQuery);

 if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
  return ERR_ORDERSTATUS_MISSING_DID_KEY;
 if ( !GetNumeric(ptr, &pOrderStatusData->d_id) )
  return ERR_ORDERSTATUS_DID_INVALID;

 if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
  return ERR_ORDERSTATUS_MISSING_CID_KEY;

 if ( *ptr == '&' || !(*ptr))
 {
  pSzTmp = szTmp;
  pOrderStatusData->c_id = 0;
```

```
  if ( !GetValuePtr(pProcessedQuery, CLT_O, &ptr) )
    return ERR_ORDERSTATUS_MISSING_CLT_KEY;
  while(*ptr != '&' && *ptr)
  {
    *pSzTmp = *ptr;
    pSzTmp++;
    ptr++;
  }
  *pSzTmp = '\0';
  _strupr( szTmp );
  strcpy(pOrderStatusData->c_last, szTmp);
  if ( strlen(pOrderStatusData->c_last) > 16 )
    return ERR_ORDERSTATUS_CLT_RANGE;
 }
 else
 {
  if (!GetNumeric(ptr, &pOrderStatusData->c_id))
    return ERR_ORDERSTATUS_CID_INVALID;
  if ( !GetValuePtr(pProcessedQuery, CLT_O, &ptr) )
    return ERR_ORDERSTATUS_MISSING_CLT_KEY;
  if ( *ptr != '&' && *ptr)
    return ERR_ORDERSTATUS_CID_AND_CLT;
 }

 return ERR_SUCCESS;
}



int ParsePaymentQuery(char *pQueryString, PaymentData *pPaymentData)
{
 char          szTmp[26];
 char          *ptr;
 char    *pPtr;
 char          *pSzTmp;
 char          *pProcessedQuery[MAXPAYMENTVALS];


 PARSE_QUERY_STRING(pQueryString, MAXPAYMENTVALS,
                                  paymentStrs, pProcessedQuery);

 if ( !GetValuePtr(pProcessedQuery, DID, &ptr) )
  return ERR_PAYMENT_MISSING_DID_KEY;
 if ( !GetNumeric(ptr, &pPaymentData->d_id) )
  return ERR_PAYMENT_DISTRICT_INVALID;

 if ( !GetValuePtr(pProcessedQuery, CID, &ptr) )
  return ERR_PAYMENT_MISSING_CID_KEY;

 pPaymentData->c_id = 0;
 if ( (*ptr != '&' && *ptr) && !GetNumeric(ptr, &pPaymentData->c_id) )
  return ERR_PAYMENT_CUSTOMER_INVALID;


 if(*ptr == '&' || !(*ptr))
 {
  pSzTmp = szTmp;
  if ( !GetValuePtr(pProcessedQuery, CLT_P, &ptr) )
    return ERR_PAYMENT_MISSING_CLT;
  while(*ptr != '&' && *ptr)
  {
    *pSzTmp = *ptr;
    pSzTmp++;
    ptr++;
  }
  *pSzTmp = '\0';
  _strupr( szTmp );

  strcpy(pPaymentData->c_last, szTmp);
  if ( strlen(pPaymentData->c_last) > 16 )
    return ERR_PAYMENT_LAST_NAME_TO_LONG;
 }
 else
 {
  if ( !GetValuePtr(pProcessedQuery, CLT_P, &ptr) )
    return ERR_PAYMENT_MISSING_CLT_KEY;
  if(*ptr != '&' && *ptr)
    return ERR_PAYMENT_CID_AND_CLT;
 }

 if ( !GetValuePtr(pProcessedQuery, CDI, &ptr) )
  return ERR_PAYMENT_MISSING_CDI_KEY;
 if ( !GetNumeric(ptr, &pPaymentData->c_d_id) )
  return ERR_PAYMENT_CDI_INVALID;

 if ( !GetValuePtr(pProcessedQuery, CWI, &ptr) )
  return ERR_PAYMENT_MISSING_CWI_KEY;

 if ( !GetNumeric(ptr, &pPaymentData->c_w_id) )
  return ERR_PAYMENT_CWI_INVALID;



 if ( !GetValuePtr(pProcessedQuery, HAM, &ptr) )
  return ERR_PAYMENT_MISSING_HAM_KEY;

 pPtr = ptr;
 while( *pPtr != '&' && *pPtr)
 {
  if ( *pPtr == '.' )
  {
    pPtr++;
    if ( !*pPtr )
          break;
    if ( *pPtr < '0' || *pPtr > '9' )
          return ERR_PAYMENT_HAM_INVALID;
    pPtr++;
    if ( !*pPtr )
          break;
    if ( *pPtr < '0' || *pPtr > '9' )
          return ERR_PAYMENT_HAM_INVALID;
    if ( !*pPtr )
          return ERR_PAYMENT_HAM_INVALID;
  }
  else if ( *pPtr < '0' || *pPtr > '9' )
    return ERR_PAYMENT_HAM_INVALID;
  pPtr++;
 }

 pPaymentData->h_amount = atof(ptr);
 if ( pPaymentData->h_amount >= 10000.00 || pPaymentData->h_amount < 0 )
  return ERR_PAYMENT_HAM_RANGE;

 return ERR_SUCCESS;
}



int ReadRegistrySettings(void)
{
 HKEY      hKey;
 DWORD   size;
 DWORD   type;
 char          szTmp[FILENAMESIZE];
 int   status;
 int           iTmp;


 status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC",
                                  0, KEY_READ, &hKey);
 if( status != ERROR_SUCCESS )
  return ERR_CANT_FIND_TPCC_KEY;
```

```
  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    return ERR_CANT_FIND_PATH_VALUE;
  strcpy(szTpccLogPath, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Server", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_SERVER_VALUE;
  strcpy(gszServer, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Database", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_DATABASE_VALUE;
  strcpy(gszDatabase, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "User", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_USER_VALUE;
  strcpy(gszUser, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "Password", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )

    return ERR_CANT_FIND_PASSWORD_VALUE;
  strcpy(gszPassword, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "CkptUser", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    strcpy(gszCkptUser, gszUser);
  else
    strcpy(gszCkptUser, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "CkptPassword", 0, &type, szTmp, &size);
  if ( status != ERROR_SUCCESS )
    strcpy(gszCkptPassword, gszPassword);
  else
    strcpy(gszCkptPassword, szTmp);

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "LOG", 0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS  && 0 == stricmp(szTmp, "ON") )
    bLog = TRUE;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "MaximumWarehouses", 0, &type, szTmp,
&size);
  if ( status == ERROR_SUCCESS  && 0 != (iTmp = atoi(szTmp)) )
    iMaxWareHouses = iTmp;

  size = sizeof(szTmp);
  status = RegQueryValueEx(hKey, "MaxConnections", 0, &type, szTmp, &size);
  if ( status == ERROR_SUCCESS && 0 != (iTmp = atoi(szTmp)) )
    iMaxConnections = iTmp;

  RegCloseKey(hKey);

  if( RegOpenKeyEx(HKEY_LOCAL_MACHINE,

"SYSTEM\\CurrentControlSet\\Services\\InetInfo\\Parameters",
                                0, KEY_READ, &hKey) != ERROR_SUCCESS )
    return ERR_CANT_FIND_INETINFO_KEY;

  size = sizeof(gdwPoolThreadLimit);
  if ( RegQueryValueEx(hKey, "PoolThreadLimit", 0, &type,
                                (LPBYTE)&gdwPoolThreadLimit, &size) !=
ERROR_SUCCESS )
    return ERR_CANT_FIND_POOLTHREADLIMIT;

  RegCloseKey(hKey);

  return ERR_SUCCESS;

}
```

tpcc.h

```
#ifndef TPCC_H
#define TPCC_H

/*+**************************************************************************
 *                                                                         *
 * COPYRIGHT (c) 1997 BY                                                   *
 * COMPAQ COMPUTER CORPORATION, .                                          *
 * ALL RIGHTS RESERVED.                                                    *
 *                                                                         *
 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
 * ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
 * OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
 * TRANSFERRED.                                                            *
 *                                                                         *
 * THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
 * AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY COMPAQ COMPUTER     *
 * CORPORATION.                                                            *
 *                                                                         *
 * COMPAQ ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY COMPAQ.                  *
 *                                                                         *
 *                                                                         *
 **************************************************************************/

/*+
 * Abstract: This is the header file for web_ui.c.  it contains the
 *                function prototypes for the routines that are called outside web_ui.c
 *
 * Author: A Bradley
 * Creation Date: May 1997
 *
 * Modified history:
 *
 *
 */


#define FILENAMESIZE 256

#if defined WEB_UI_C || defined TPCC_C

void FormatString(char *szDest, char *szPic, char *szSrc);

int ParseNewOrderQuery(char *pQueryString, NewOrderData
*pNewOrderData);
int ParsePaymentQuery(char *pQueryString, PaymentData *pPaymentData);
int ParseOrderStatusQuery(char *pQueryString,
                                        OrderStatusData *pOrderStatusData);
#endif

BOOL ReadRegistrySettings(void);


#ifdef TPCC_C
#define GLOBAL(thing,initializer) thing = initializer
#else
#define GLOBAL(thing,initializer) extern thing
```

```
#endif

#if defined WEB_UI_C || defined TPCC_C
#endif
GLOBAL(int iMaxConnections,25);
GLOBAL(BOOL bLog,FALSE);
GLOBAL(int iDeadlockRetry,3);
GLOBAL(char szTpccLogPath[FILENAMESIZE],{'\0'});
GLOBAL(int iMaxWareHouses,500);
GLOBAL(char gszServer[32],{'\0'});
GLOBAL(char gszDatabase[32],"tpcc");
GLOBAL(char gszUser[32],"sa");
GLOBAL(char gszPassword[32],{'\0'});
GLOBAL(char gszCkptUser[32],"sa");
GLOBAL(char gszCkptPassword[32],{'\0'});
GLOBAL(pTransactionPoolStruct gpTransactionPool,{0});

#endif
```

tpcc_gen.c

```
/*+
21-Feb-97   Ruth                    Merged unix_gen with oe_gen to make
single file
17-Jan-97   WCarr                   Added new format header.  Moved
transaction
                                            definition flags to common
header so they
                                            could be shared by new run-
time data
                                            reduction script.  Modified
code to add
                                            name of CGI script (dll)
dynamically.
 *
 * Copyright 1997 (c) Compaq Computer Corporation
 *
 * Description:
 *
 * These are the five routines to be called by the VAXRTE scripts to generate
 * input data for the five different TPC-C transaction types.  All input data
 * are generated based on the requirement of the TPC-C spec.
 *
 ******************************************************************************
-*/

#include <stdio.h>
#include <math.h>
#include <string.h>
#include <time.h>

#ifdef _WIN32
# include <nt_lib.h>
#endif

#include <stdtypes.h>
#include <common.h>
#include <prte.h>
#include <tpcc_gen.h>



static char *no_ol_strs[] = {
"A=%d&B=%d&C=%d&",
"D=%d&E=%d&F=%d&",
"G=%d&H=%d&I=%d&",
"J=%d&K=%d&L=%d&",
"M=%d&N=%d&O=%d&",
"P=%d&Q=%d&R=%d&",
"S=%d&T=%d&U=%d&",
"V=%d&W=%d&X=%d&",
"a=%d&b=%d&c=%d&",
"d=%d&e=%d&f=%d&",
"g=%d&h=%d&i=%d&",
"j=%d&k=%d&l=%d&",
"m=%d&n=%d&o=%d&",
"p=%d&q=%d&r=%d&",
"s=%d&t=%d&u=%d&",
};

static char *no_blank_ol_strs[] = {
"A=&B=&C=&",
"D=&E=&F=&",
"G=&H=&I=&",
"J=&K=&L=&",
"M=&N=&O=&",
"P=&Q=&R=&",
"S=&T=&U=&",
"V=&W=&X=&",
"a=&b=&c=&",
"d=&e=&f=&",
"g=&h=&i=&",
"j=&k=&l=&",
"m=&n=&o=&",
"p=&q=&r=&",
"s=&t=&u=&",
};


#define C_ID_CONST 511
#define ITEM_CONST 4093



#define PCT_C_LAST 60
#define ILLEGAL_ITEM    111111


#define NURand(A, x, y, C) \
    ((((random_int(0, A) | random_int(x, y)) + C) % (y-x+1)) + x)


const char *c_table[10] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES", "ESE",
                    "ANTI", "CALLY", "ATION", "EING"};
const int c_length[10] = {3,5,4,3,4,3,4,5,5,4};

void
get_no_data(user_context_t *pContext, char *string, int *context)
{
 int i, remote_txn=0, length=0, d_id, c_id;
 char temp_buf[200];

 char *init_get="GET %s?3=N%d&8=%d&9=%d&";
 int current_line=0;

 long int ol_supply_w_id, ol_quantity, ol_item_id, ol_count;

 *context = NEW_ORDER_TXN;


 d_id = random_int(1,10);

 c_id = NURand(1023, 1, 3000, C_ID_CONST );

 sprintf(string, init_get, pContext->cgi_script_name,
                WDID(pContext->w_id,pContext->d_id), d_id, c_id);

 ol_count = random_int(5, 15 );
 *context |= (ol_count << 4);
 for (i=0; i < ol_count; i++) {

  if ((random_int(0,999 ) < 990) || (pContext->w_max == 1))
   ol_supply_w_id = pContext->w_id;
```

```
   else {
     remote_txn++;
     if ((ol_supply_w_id = random_int(1, pContext->w_max-1 )) >= pContext-
>w_id)
                ol_supply_w_id++;
   }



   if (i == ol_count-1) {
     if (random_int(0,999 ) < 10) {
                ol_item_id = ILLEGAL_ITEM;
                *context |= ILLEGAL_ITEM_TXN;
      }
      else
                ol_item_id = NURand(8191, 1, 100000, ITEM_CONST );
   }
   else
     ol_item_id = NURand(8191, 1, 100000, ITEM_CONST );

   ol_quantity = random_int(1, 10 );

   sprintf(temp_buf, no_ol_strs[current_line],
                   ol_supply_w_id, ol_item_id, ol_quantity);

   strcat(string,temp_buf);
   current_line++;
  }

  for(i=current_line; i < 15 ; i++)
   {
    sprintf(temp_buf, no_blank_ol_strs[i]);
    strcat(string,temp_buf);
   }
  strcat(string, "0=Process HTTP/1.0\nConnection: Keep-Alive\n\n");


 *context |= (remote_txn << 8);
}

void
get_pt_data(user_context_t *pContext, char *string, int *context)
{
  char *ptr;
  char name[20];
  int length = 0;
  int index, i1, i2, i3;
  int c_id=-1;
  int  c_w_id, c_d_id, d_id;
  int h_amount;
  float flt_h_amount;

  *context = PAYMENT_TXN;

  d_id = random_int(1, 10 );

  ptr = name;
  if (random_int(0,99 ) < PCT_C_LAST) {
   *context |= NON_KEY_ACCESS;
   index = NURand(255, 0, 999, pContext->CLast );
   i1 = index/100;
   i2 = (index - i1 * 100) / 10;
   i3 = index - (i1 * 100) - (i2 * 10);


   memcpy(ptr, c_table[i1], c_length[i1]);
   ptr += c_length[i1];
   memcpy(ptr, c_table[i2], c_length[i2]);
   ptr += c_length[i2];
   memcpy(ptr, c_table[i3], c_length[i3]);
   ptr += c_length[i3];
   *ptr = '\0';
  }
  else {
    c_id = NURand(1023, 1, 3000, C_ID_CONST );

  }

  if ((random_int(0,999 ) < 850) || (pContext->w_max == 1)) {
    c_w_id = pContext->w_id;
    c_d_id = d_id;
  }
  else {
    *context |= REMOTE_PAYMENT;
    if ((c_w_id= random_int(0,pContext->w_max-2 )+1) >= pContext->w_id)
      c_w_id++;
    c_d_id = random_int(1, 10 );
  }



  h_amount = random_int(100, 500000 );

  flt_h_amount = h_amount/100.00f;
  if ( -1 == c_id)
    sprintf(string,"GET
%s?3=P%d&8=%d&9=&Z=%d&v=%d&Y=%s&w=%f&0=Process "
                   "HTTP/1.0\nConnection: Keep-Alive\n\n",
                   pContext->cgi_script_name,
                   WDID(pContext->w_id,pContext->d_id),
                   d_id, c_w_id, c_d_id, name, flt_h_amount);
  else
    sprintf(string,"GET
%s?3=P%d&8=%d&9=%d&Z=%d&v=%d&Y=&w=%f&0=Process "
                   "HTTP/1.0\nConnection: Keep-Alive\n\n",
                   pContext->cgi_script_name,
                   WDID(pContext->w_id,pContext->d_id),
                   d_id, c_id, c_w_id, c_d_id, flt_h_amount);
}

void
get_os_data(user_context_t *pContext, char *string, int *context)
{
  char *ptr;
  char name[20];
  int c_id=-1;
  int length = 0, index, i1, i2, i3, d_id;

  *context = ORDER_STATUS_TXN;

  d_id = random_int(1, 10 );

  ptr=NULL;
  if (random_int(0,99 ) < PCT_C_LAST) {
    ptr=name;
    *context |= NON_KEY_ACCESS;
    index = NURand(255, 0, 999, pContext->CLast );
    i1 = index/100;
    i2 = (index - i1 * 100) / 10;
    i3 = index - (i1 * 100) - (i2 * 10);

    memcpy(ptr, c_table[i1], c_length[i1]);
    ptr += c_length[i1];
    memcpy(ptr, c_table[i2], c_length[i2]);
    ptr += c_length[i2];
    memcpy(ptr, c_table[i3], c_length[i3]);
    ptr += c_length[i3];
    *ptr = '\0';
  }
  else {
    c_id = NURand(1023, 1, 3000, C_ID_CONST );

  }
```

```
 if( -1 == c_id)
   sprintf(string, "GET %s?3=O%d&8=%d&9=&Y=%s&0=Process "
                   "HTTP/1.0\nConnection: Keep-Alive\n\n",
                   pContext->cgi_script_name,
                   WDID(pContext->w_id,pContext->d_id),
                   d_id, name);
 else
   sprintf(string, "GET %s?3=O%d&8=%d&9=%d&Y=&0=Process "
                   "HTTP/1.0\nConnection: Keep-Alive\n\n",
                   pContext->cgi_script_name,
                   WDID(pContext->w_id,pContext->d_id),
                   d_id, c_id);
}

void
get_dy_data(user_context_t *pContext, char *string, int *context)
{
 int carrier_id;
 *context = INT_DELIVERY_TXN;

 carrier_id = random_int(1,10);
 sprintf(string, "GET %s?3=D%d&6=%d&7=%d&0=Process "
                 "HTTP/1.0\nConnection: Keep-Alive\n\n",
                 pContext->cgi_script_name,
                 WDID(pContext->w_id,pContext->d_id),
                 time(NULL), carrier_id);
}

void
get_sl_data(user_context_t *pContext, char *string, int *context)
{
 int threshold;

 *context = STOCK_LEVEL_TXN;

 threshold = random_int(10,20);
 sprintf(string, "GET %s?3=S%d&x=%d&0=Process "
                 "HTTP/1.0\nConnection: Keep-Alive\n\n",
                 pContext->cgi_script_name,
                 WDID(pContext->w_id,pContext->d_id),
                 threshold);
}
```

tpcc_gen.h

```
#ifndef TPCC_GEN_H
#define TPCC_GEN_H

#include <tpcc_user.h>

void get_no_data (user_context_t *pUdata, char *string, int *context);
void get_pt_data (user_context_t *pUdata, char *string, int *context);
void get_os_data (user_context_t *pUdata, char *string, int *context);
void get_dy_data (user_context_t *pUdata, char *string, int *context);
void get_sl_data (user_context_t *pUdata, char *string, int *context);

#endif
```

tpcc_master.c

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <sys/stat.h>
#include <sys/types.h>
#include <time.h>

#ifdef _WIN32
# include <direct.h>
# include <nt_lib.h>
# include <windows.h>
# include <process.h>
#else
# include <dirent.h>
# include <sys/time.h>
# include <unistd.h>
#endif



#include <common.h>
#include <prte.h>
#include <msg.h>
#include <scr_util.h>
#include <audit_util.h>
#include <stdtypes.h>
#include <tpcc_master.h>
#include <netvar.h>


#ifdef _WIN32
# define MKDIR(Dir,Mode) mkdir(Dir)
  typedef int mode_t;
#else
# define MKDIR(Dir,Mode) mkdir(Dir,Mode)
#endif

#define ERR_SUCCESS 0

#define MAX_RUN_DIR_NAME_LEN 3
#define LOG_DIR "logs"
#define maximum(A,B) ((A) > (B) ? (A) : (B))

#define NODELAY

#define SHELL_PROMPT_PART_1 ("CONTROL")
#define SHELL_PROMPT_PART_2 (":")
#define SHELL_PROMPT       ("CONTROL:")

#define LOGIN_PROMPT  ("login:")
#define PASSWD_PROMPT ("Password:")


#define ORACLE 0
#define SYBASE 1
#define MSSQL 2



void        BinMsgHandler( void *Context, int SendersID, int Len, void *Msg
);
int32       GetBeInfo(master_data_t *pContext);
void        GetFeUserCounts(master_data_t *pContext, int32 State);
void        GetSeed(master_data_t *pContext);
int32       Init(master_data_t *pContext);
int32       MakeRunDir(char *DirName, char *RunDir);
int32       ParsePerfCounterPage(web_page_t *Page);
int32       ParseUsersPerFE(master_data_t *pContext, char *string);
int32   PreTestHook(master_data_t *pContext);
int         SCR_count_elements (char *List);
char        *SCR_next_list_element (char *List, char *Element);
void        TellAidesTo(master_data_t *pContext, int32 Task, int32 State);



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

```
checkpoint_init_msg_t       CheckpointInit;
checkpoint_run_msg_t        CheckpointRun;
master_data_t                       Context;
master_data_t                       *pContext = &Context;
measurement_info_msg_t      Measurement;
int32                               Index;
generic_msg_t                       GenericMsg;
double                      TimeStamp;
double                      TimeStamp2;
double                      DelayTime;
double                      StartTime;



PRTEinit( argc, argv );
SCRlog( "\n\n\n\t\t\t\t\t..START OF TEST..\n\n" );


if( ERR_SUCCESS != Init( pContext ))
{
 ( void ) SCRerr( ERR_INIT );
 PRTEexit();
}



SCRlog( "\tReducer being told to initialize.\n\n" );
pContext->ReducerFailed = FALSE;
pContext->MsgTimedOut = TRUE;
PRTEset_network_variable( "INIT_REDUCER", "TRUE" );
PRTEdelay( pContext->MsgTimeout );

SCRlog( "Master: Resuming.\n" );

if( pContext->MsgTimedOut )
{

 SCRerr( ERR_REDUCER_INIT_TIMEOUT );
 SCRlog( "\tPausing indefinitely....\n\n" );
 PRTEpause();
}

if( pContext->ReducerFailed )
{
 SCRlog( "\tReducer failed to initialize correctly. Pausing.\n" );
 PRTEpause();
}

SCRlog( "\tReducer successfully initialized.\n" );



if( 0 < pContext->AideCount )
{
 SCRlog( "\t%d Aides being told to initialize.", pContext->AideCount );
 pContext->RequestsSent = pContext->AideCount;
 pContext->ReplysReceived = 0;
 pContext->AidesFailed = FALSE;
 pContext->MsgTimedOut = TRUE;
 PRTEset_network_variable( "INIT_AIDES", "TRUE" );
 PRTEdelay( pContext->MsgTimeout );

 if( pContext->MsgTimedOut )
 {

  ( void ) SCRerr( ERR_AIDE_INIT_TIMEOUT );
  SCRlog( "\tPausing indefinitely....\n\n" );
  PRTEpause();
 }

 if( pContext->AidesFailed )
 {
  SCRlog( "\tNot all aides initialized correctly. Pausing.\n" );
  PRTEpause();
 }
 SCRlog( "\t%d Aides successfully initialized.\n", pContext->AideCount );
}



 if( RTE_LOOPBACK_MODE != pContext->LoopbackMode )
 {
  SCRlog( "\tUser DLL(s) being loaded." );
  pContext->ReducerFailed = FALSE;
  pContext->MsgTimedOut = TRUE;
  GenericMsg.Type = LOAD_USER_DLL_REQUEST_MSG;
  PRTEmessage_to_user_binary( sizeof( GenericMsg ), &GenericMsg,
                                                    pContext->ReducerID,
pContext->ReducerID );
  PRTEdelay( pContext->MsgTimeout );

  if( pContext->MsgTimedOut )
  {

   SCRerr( ERR_LOAD_USER_DLL_TIMEOUT );
   SCRlog( "\tPausing indefinitely....\n\n" );
   PRTEpause();
  }

  if( pContext->ReducerFailed )
  {
   SCRlog( "\tUser DLL(s) failed to load correctly. Pausing.\n" );
   PRTEpause();
  }

  SCRlog( "\tUser DLL(s) successfully loaded.\n" );

 }


 if(( END_TO_END_MODE == pContext->LoopbackMode ) && pContext-
>DoCheckpoints )
 {
  TellAidesTo( pContext, BE_CKPTDATA_REQUEST_MSG, BEFORE );
 }


 if(( END_TO_END_MODE == pContext->LoopbackMode ) &&
   pContext->DbType == ORACLE)
 {
  TellAidesTo( pContext, ORA_SWITCHLOG_REQUEST_MSG, BEFORE );

  if( '\0' != pContext->BeTasksData.OraStatScriptPath[0] )
  {
   TellAidesTo( pContext, ORA_DOSTATS_REQUEST_MSG, BEFORE );
  }
 }


 if( pContext->BeTasksData.GetAllAuditFiles )
 {
  TellAidesTo( pContext, INIT_AUDIT_TABLE_REQUEST_MSG, BEFORE
);
  TellAidesTo( pContext, GET_LOGSIZE_REQUEST_MSG, BEFORE );
 }


 if( '\0' != pContext->BeTasksData.CustomBeforeTestScript[0] )
 {
  TellAidesTo( pContext, DO_CUSTOM_SCRIPT_REQUEST_MSG, BEFORE
);
 }
```

```
 SCRlog("\tPausing while users log in.\n\n");
 pContext->ReducerFailed = FALSE;
 PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
 PRTEpause();


 SCRlog("Master: Resuming.\n");

 if( pContext->ReducerFailed )
 {
  SCRlog( "\tUsers failed to log in successfully. Pausing.\n" );
  PRTEpause();
 }
 else
 {
  SCRlog( "\tAll users logged in successfully.\n" );
 }



 if( RTE_LOOPBACK_MODE != pContext->LoopbackMode )
 {
  SCRlog( "\tGetting user counts on all FEs." );
  pContext->ReducerFailed = FALSE;
  pContext->MsgTimedOut = TRUE;
  GenericMsg.Type = GET_USER_COUNTS_REQUEST_MSG;
  PRTEmessage_to_user_binary( sizeof( GenericMsg ), &GenericMsg,
                                                    pContext->ReducerID,
pContext->ReducerID );
  PRTEdelay( pContext->MsgTimeout );

  if( pContext->MsgTimedOut )
  {

   SCRerr( ERR_USER_COUNTS_TIMEOUT );
   SCRlog( "\tPausing indefinitely....\n\n" );
   PRTEpause();
  }

  if( pContext->ReducerFailed )
  {
   SCRlog( "\tFailed to get user counts on all FEs. Pausing.\n" );
   PRTEpause();
  }

  SCRlog( "\tSuccessfully got all user counts.\n" );
 }


 if( pContext->DoCheckpoints )
 {
  SCRlog( "\t%d Aides initializing checkpoint sub-systems.",
                pContext->AideCount );
  CheckpointInit.Type = CHECKPOINT_INIT_REQUEST_MSG;
  strcpy( CheckpointInit.Data.Dll, pContext->CgiScriptName );
  pContext->RequestsSent = pContext->AideCount;
  pContext->ReplysReceived = 0;
  pContext->AidesFailed = FALSE;
  pContext->MsgTimedOut = TRUE;

  for( Index = 0; Index < pContext->AideCount; Index++ )
  {
   strcpy( CheckpointInit.Data.FEName, pContext->pAdminFEs[Index].Name
);
   PRTEmessage_to_user_binary( sizeof( CheckpointInit ), &CheckpointInit,
                                                    pContext-
>pAides[Index].PrteID,
                                                    pContext-
>pAides[Index].PrteID );

  }
  PRTEdelay( pContext->MsgTimeout );

  if( pContext->MsgTimedOut )
  {
   ( void ) SCRerr( ERR_AIDE_CKPT_INIT_TIMEOUT );
   SCRlog( "Pausing indefinitely...\n\n" );
   PRTEpause();
  }

  if( pContext->AidesFailed )
  {
   SCRlog( "Not all aides did checkpoint initialization successfully." );
   SCRlog( "Pausing...\n\n" );
   PRTEpause();
  }

  SCRlog( "\t%d Aides successfully initialized checkpoint sub-systems.\n",
                pContext->ReplysReceived );
 }



 SCRlog("\tPausing while users start doing transactions.\n\n");
 pContext->ReducerFailed = FALSE;
 PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
 PRTEpause();


 SCRlog("Master: Resuming.\n");

 if( pContext->ReducerFailed )
 {
  SCRlog( "\tUsers failed to resume successfully. Pausing.\n" );
  PRTEpause();
 }
 else
 {
  SCRlog( "\tAll users starting transactions successfully.\n" );
 }


 TimeStamp = PRTEtimer_begin( "x" );
 PRTEtimer_cancel();

 SCRlog( "\n\n\n\n\t\t\t\tStart of Warmup\n\n");


 if( pContext->DoCheckpoints )
 {
  SCRlog( "\t%d Aides starting checkpoint loop.",
                pContext->AideCount );



  CheckpointRun.Type = CHECKPOINT_RUN_REQUEST_MSG;
  StartTime = TimeStamp + pContext->CkptStartOffset;
  CheckpointRun.Data.StartTime = StartTime;
  CheckpointRun.Data.Interval = pContext->CheckpointInterval;
  strcpy( CheckpointRun.Data.Dll, pContext->CgiScriptName );

  pContext->RequestsSent = pContext->AideCount;
  pContext->ReplysReceived = 0;
  pContext->AidesFailed = FALSE;
  pContext->MsgTimedOut = TRUE;

  for( Index = 0; Index < pContext->AideCount; Index++ )
  {
   strcpy( CheckpointRun.Data.FEName, pContext->pAdminFEs[Index].Name
);
   PRTEmessage_to_user_binary( sizeof( CheckpointRun ), &CheckpointRun,
                                                    pContext-
>pAides[Index].PrteID,
```

```
                                                pContext-
>pAides[Index].PrteID );

    }

    PRTEdelay( pContext->MsgTimeout );

    if( pContext->MsgTimedOut )
    {
      ( void ) SCRerr( ERR_AIDE_CKPT_RUN_TIMEOUT );
      SCRlog( "Pausing indefinitely...\n\n" );
      PRTEpause();
    }

    if( pContext->AidesFailed )
    {
      SCRlog( "Not all aides entered checkpoint loop.." );
      SCRlog( "Pausing...\n\n" );
      PRTEpause();
    }

    SCRlog( "\t%d Aides successfully entered checkpoint loop.\n",
                    pContext->ReplysReceived );

  }



  TimeStamp2 = PRTEtimer_begin( "x" );
  PRTEtimer_cancel();
  DelayTime = pContext->WarmUpTime - (TimeStamp2 - TimeStamp);

  PRTEdelay( DelayTime );


  TimeStamp = PRTEtimer_begin( "x" );
  PRTEtimer_cancel();

  SCRlog( "\n\n\n\n\t\t\t\tStart of Measurement\n\n" );

  Measurement.Type = MEASUREMENT_INFO_MSG;
  Measurement.Data.Length = pContext->SteadyStateTime;
  Measurement.Data.StartTime = TimeStamp;
  PRTEmessage_to_user_binary(sizeof(Measurement), &Measurement,
                                        pContext->ReducerID, pContext-
>ReducerID);

  TimeStamp2 = PRTEtimer_begin( "x" );
  PRTEtimer_cancel();
  DelayTime = pContext->SteadyStateTime - (TimeStamp2 - TimeStamp);
  PRTEdelay( DelayTime );



  TimeStamp = PRTEtimer_begin( "x" );
  PRTEtimer_cancel();

  SCRlog( "\n\n\n\n\t\t\t\tStart of Cooldown" );

  TimeStamp2 = PRTEtimer_begin( "x" );
  PRTEtimer_cancel();
  DelayTime = pContext->CoolDownTime - (TimeStamp2 - TimeStamp);
  PRTEdelay( DelayTime );

  SCRlog( "\n\n\n\n\t\t\t\tEnd of Cooldown\n\n");


  if( pContext->DoCheckpoints )
  {
    SCRlog( "\t%d Aides stopping checkpoint loop.",
                                        pContext->AideCount );
    GenericMsg.Type = CHECKPOINT_STOP_REQUEST_MSG;

    pContext->RequestsSent = pContext->AideCount;
    pContext->ReplysReceived = 0;
    pContext->AidesFailed = FALSE;
    pContext->MsgTimedOut = TRUE;

    for( Index = 0; Index < pContext->AideCount; Index++ )
    {
      PRTEmessage_to_user_binary( sizeof( GenericMsg ), &GenericMsg,
                                                pContext-
>pAides[Index].PrteID,
                                                pContext-
>pAides[Index].PrteID );
    }

    PRTEdelay( pContext->MsgTimeout );

    if( pContext->MsgTimedOut )
    {
      ( void ) SCRerr( ERR_AIDE_CKPT_STOP_TIMEOUT );
      SCRlog( "Pausing indefinitely...\n\n" );
      PRTEpause();
    }

    if( pContext->AidesFailed )
    {
      SCRlog( "Not all aides successfully ran checkpoint loop." );
      SCRlog( "Pausing...\n\n" );
      PRTEpause();
    }

    SCRlog( "\t%d Aides successfully ran checkpoint loop.\n",
                    pContext->ReplysReceived );

  }


  PRTEstop_users( pContext->ReducerID, pContext->ReducerID );
  PRTEpause();


  if( RTE_LOOPBACK_MODE != pContext->LoopbackMode )
  {
    SCRlog( "\tGetting user counts on all FEs." );
    pContext->ReducerFailed = FALSE;
    pContext->MsgTimedOut = TRUE;
    GenericMsg.Type = GET_USER_COUNTS_REQUEST_MSG;
    PRTEmessage_to_user_binary( sizeof( GenericMsg ), &GenericMsg,
                                                pContext->ReducerID,
pContext->ReducerID );
    PRTEdelay( pContext->MsgTimeout );

    if( pContext->MsgTimedOut )
    {

      SCRerr( ERR_USER_COUNTS_TIMEOUT );
      SCRlog( "\tPausing indefinitely....\n\n" );
      PRTEpause();
    }

    if( pContext->ReducerFailed )
    {
      SCRlog( "\tFailed to get user counts on all FEs. Pausing.\n" );
      PRTEpause();
    }

    SCRlog( "\tSuccessfully got all user counts.\n" );
  }


  SCRlog("\tPausing while users stop doing transactions.\n\n");
```

```
 pContext->ReducerFailed = FALSE;
 PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
 PRTEpause();

 SCRlog("Master: Resuming.\n");

 if( pContext->ReducerFailed )
 {
  SCRlog( "\tUsers failed to stop successfully. Pausing.\n" );
  PRTEpause();
 }
 else
 {
  SCRlog( "\tAll users successfully stopped doing transactions.\n" );
 }



 SCRlog("\tPost process binary logfile.\n\n");

 pContext->ReducerFailed = FALSE;
 PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
 PRTEpause();

 SCRlog("Master: Resuming.\n");


 if(( END_TO_END_MODE == pContext->LoopbackMode ) && pContext-
>DoCheckpoints )
   TellAidesTo( pContext, BE_CKPTDATA_REQUEST_MSG, AFTER );

 if( pContext->BeTasksData.GetAllAuditFiles )
 {
  TellAidesTo( pContext, GET_LOGSIZE_REQUEST_MSG, AFTER );
 }

 if(( END_TO_END_MODE == pContext->LoopbackMode ) &&
    pContext->DbType == ORACLE )
 {
  if( pContext->BeTasksData.OraStatScriptPath[0] )
     TellAidesTo( pContext, ORA_DOSTATS_REQUEST_MSG, AFTER );

  TellAidesTo( pContext, ORA_SWITCHLOG_REQUEST_MSG, AFTER );
 }



 if( pContext->BeTasksData.CustomAfterTestScript[0] )
   TellAidesTo( pContext, DO_CUSTOM_SCRIPT_REQUEST_MSG, AFTER
);


 SCRlog("\tPost process aide logfiles.\n\n");

 pContext->ReducerFailed = FALSE;
 PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
 PRTEpause();

 SCRlog("Master: Resuming.\n");




 if( 0 < pContext->AideCount )
 {
  SCRlog( "\t%d Aides being told to exit.", pContext->AideCount );
  pContext->RequestsSent = pContext->AideCount;
  GenericMsg.Type = AIDE_EXIT_REQUEST_MSG;
  pContext->ReplysReceived = 0;
  pContext->AidesFailed = FALSE;
  pContext->MsgTimedOut = TRUE;

   for( Index = 0; Index < pContext->AideCount; Index++ )
   {
    PRTEmessage_to_user_binary( sizeof( GenericMsg ), &GenericMsg,
                                                    pContext-
>pAides[Index].PrteID,
                                                    pContext-
>pAides[Index].PrteID );

   }

   PRTEdelay( pContext->MsgTimeout );

   if( pContext->MsgTimedOut )
   {
    ( void ) SCRerr( ERR_AIDE_EXIT_TIMEOUT );
   }

   if( pContext->AidesFailed )
   {
    SCRlog( "Not all aides exited successfully." );
   }

   SCRlog( "\t%d Aides exited successfully.\n",
                pContext->ReplysReceived );
 }

 SCRlog( "Master: Blowing this popsicle stand." );

 SCRlog( "\n\n\n\n\t\t\t\t..END OF TEST..\n\n\n" );

 PRTEexit();
}


void
BinMsgHandler( void *Context, int SendersID, int Len, void *Msg )
{
 checkpoint_status_msg_t        *pCheckpointStatusMsg;
 master_data_t                  *pContext;
 generic_msg_t                  *pMsg;
 generic_status_msg_t           *pStatusMsg;
 char                           TmpStr[256];


 pMsg = Msg;
 pContext = Context;


 switch( pMsg->Type )
 {
  case DATA_PROCESSING_LOOP_OVER:
   PRTEresume_users( User_id, User_id );
   break;

  case DATA_POST_PROCESS_REPLY_MSG:
   pStatusMsg = (generic_status_msg_t *) Msg;
   if( ERR_SUCCESS == pStatusMsg->Status )
   {
        SCRlog( "\tTest results successfully generated." );
   }
   else
   {
        SCRlog( "\tError generating test results ( %d ).",
                pStatusMsg->Status );
   }
   PRTEresume_users( User_id, User_id );

   break;


  case AIDELOG_POST_PROCESS_REPLY_MSG:
   pStatusMsg = (generic_status_msg_t *) Msg;
```

```
  if( ERR_SUCCESS == pStatusMsg->Status )
  {
        SCRlog( "\tAide logs successfully processed." );
  }
  else
  {
        SCRlog( "\tError processing aide logs( %d ).",
              pStatusMsg->Status );
  }
  PRTEresume_users( User_id, User_id );

  break;

 case AIDE_EXIT_REPLY_MSG:
  pStatusMsg = (generic_status_msg_t *) Msg;
  pContext->ReplysReceived++;
  if( ERR_SUCCESS != pStatusMsg->Status )
  {

        pContext->AidesFailed = TRUE;
  }
  if( pContext->ReplysReceived == pContext->RequestsSent )
  {


        pContext->MsgTimedOut = FALSE;
        PRTEdelay( 0.0 );
  }
  break;

 case AIDE_INIT_REPLY_MSG:
  pStatusMsg = (generic_status_msg_t *) Msg;
  if( ERR_SUCCESS == pStatusMsg->Status )
  {

        (pContext->pAides + pContext->ReplysReceived)->PrteID =
SendersID;
  }
  else
  {

        pContext->AidesFailed = TRUE;
  }
  pContext->ReplysReceived++;
  if( pContext->ReplysReceived == pContext->RequestsSent )
  {


        pContext->MsgTimedOut = FALSE;
        PRTEdelay( 0.0 );
  }
  break;

 case CHECKPOINT_INIT_REPLY_MSG:
  pCheckpointStatusMsg = (checkpoint_status_msg_t *) Msg;
  pContext->ReplysReceived++;

  if( ERR_SUCCESS == pCheckpointStatusMsg->Data.Status )
  {
        SCRlog( "\t\t%s - checkpoint sub-system initialized.",
              pCheckpointStatusMsg->Data.FEName );
  }
  else
  {
        pContext->AidesFailed = TRUE;
        SCRlog( "\t\t%s - checkpoint subsystem initialization failed ( %d ).",
              pCheckpointStatusMsg->Data.FEName,
              pCheckpointStatusMsg->Data.Status );
  }

  if( pContext->ReplysReceived == pContext->RequestsSent )
  {
        pContext->MsgTimedOut = FALSE;
        PRTEdelay( 0.0 );
  }

  break;

 case CHECKPOINT_RUN_REPLY_MSG:
  pCheckpointStatusMsg = (checkpoint_status_msg_t *) Msg;
  pContext->ReplysReceived++;

  if( ERR_SUCCESS == pCheckpointStatusMsg->Data.Status )
  {
        SCRlog( "\t\t%s - checkpoint loop started.",
              pCheckpointStatusMsg->Data.FEName );
  }
  else
  {
        pContext->AidesFailed = TRUE;
        SCRlog( "\t\t%s - checkpoint loop NOT entered ( %d ).",
              pCheckpointStatusMsg->Data.FEName,
              pCheckpointStatusMsg->Data.Status );
  }

  if( pContext->ReplysReceived == pContext->RequestsSent )
  {


        pContext->MsgTimedOut = FALSE;
        PRTEdelay( 0.0 );
  }

  break;

 case CHECKPOINT_STOP_REPLY_MSG:
  pCheckpointStatusMsg = (checkpoint_status_msg_t *) Msg;
  pContext->ReplysReceived++;

  if( ERR_SUCCESS == pCheckpointStatusMsg->Data.Status )
  {
        SCRlog( "\t\t%s - checkpoint loop ran successfully.",
              pCheckpointStatusMsg->Data.FEName );
  }
  else
  {
        pContext->AidesFailed = TRUE;
        SCRlog( "\t\t%s - checkpoint loop FAILED ( %d ).",
              pCheckpointStatusMsg->Data.FEName,
              pCheckpointStatusMsg->Data.Status );
  }

  if( pContext->ReplysReceived == pContext->RequestsSent )
  {


        pContext->MsgTimedOut = FALSE;
        PRTEdelay( 0.0 );
  }

  break;


 case ORA_SWITCHLOG_REPLY_MSG:
 case GET_LOGSIZE_REPLY_MSG:
 case ORA_DOSTATS_REPLY_MSG:
 case DO_CUSTOM_SCRIPT_REPLY_MSG:
 case INIT_AUDIT_TABLE_REPLY_MSG:
 case BE_CKPTDATA_REPLY_MSG:
  pStatusMsg = (generic_status_msg_t *) Msg;
  if( ERR_SUCCESS != pStatusMsg->Status )
  {
```

```
                pContext->AidesFailed = FALSE;
    }
    pContext->ReplysReceived++;
    if( pContext->ReplysReceived == pContext->RequestsSent )
    {


                pContext->MsgTimedOut = FALSE;
                PRTEdelay( 0.0 );
    }
    break;

   case REDUCER_INIT_REPLY_MSG:
    pStatusMsg = (generic_status_msg_t *) Msg;
    pContext->ReducerID = SendersID;
    if( ERR_SUCCESS != pStatusMsg->Status )
    {

                pContext->ReducerFailed = TRUE;
    }
    pContext->MsgTimedOut = FALSE;
    PRTEdelay( 0.0 );
    break;

   case GET_USER_COUNTS_REPLY_MSG:
    pStatusMsg = (generic_status_msg_t *) Msg;
    if( ERR_SUCCESS != pStatusMsg->Status )
    {

                pContext->ReducerFailed = TRUE;
    }
    pContext->MsgTimedOut = FALSE;
    PRTEdelay( 0.0 );
    break;

   case USER_LOGIN_REPLY_MSG:
   case USER_RESUME_REPLY_MSG:
   case USER_STOP_REPLY_MSG:
    pStatusMsg = (generic_status_msg_t *) Msg;
    if( ERR_SUCCESS != pStatusMsg->Status )
    {

                pContext->ReducerFailed = TRUE;
    }
    PRTEresume_users( User_id, User_id );
    break;

   case LOAD_USER_DLL_REPLY_MSG:
    pStatusMsg = (generic_status_msg_t *) Msg;
    if( ERR_SUCCESS != pStatusMsg->Status )
    {

                pContext->ReducerFailed = TRUE;
    }
    pContext->MsgTimedOut = FALSE;
    PRTEdelay( 0.0 );
    break;

   default:
    sprintf( TmpStr, "Master received unknown user message type: %d", *pMsg);
    SCRlog( TmpStr );
  }
}



int32
GetBeInfo( master_data_t *pContext )
{
  char          *List;
  char          *pTmpChar;
  char          ElementString[32];
  int32         BeIndex;


  if( NULL == ( List = PRTEget_network_variable( "BE_NAMES" )))
  {
   return( SCRerr( ERR_BE_NAMES ));
  }

  pContext->NumBE = SCR_count_elements(List);


  if( NULL == ( pContext->pBEs =
                                ( be_t * ) malloc( pContext->NumBE * sizeof( be_t ))))
  {
   return( SCRerr( ERR_MALLOC_BE_ARRAY ));
  }

  SCRlog( "\tThe following BEs will be used." );
  pTmpChar = List;
  BeIndex = 0;
  while( pTmpChar = SCR_next_list_element( pTmpChar, ElementString ))
  {
    pContext->pBEs[BeIndex].Name =
      ( char * ) malloc( sizeof( ElementString )+ 1 );
    strcpy( pContext->pBEs[BeIndex].Name, ElementString );
    SCRlog( "\t\t%s", pContext->pBEs[BeIndex].Name );
    BeIndex++;
  }

  return ERR_SUCCESS;
}



void
GetSeed(master_data_t *pContext)
{

  char temp_string[128];
  struct timeval current_time;
  FILE   *SeedFile;
  char   SeedFileName[256];


  SCR_getnet_long("SEED",SEED_DEFAULT,&pContext->MasterSeed);
  if (0 == pContext->MasterSeed)
  {
    gettimeofday( &current_time, NULL);
    pContext->MasterSeed = pContext->OsPid * current_time.tv_sec;
  }


  sprintf(temp_string,"%ld",pContext->MasterSeed);
  PRTEset_network_variable("SEED",temp_string);


  sprintf(SeedFileName, "%sm%sseed.v%d", pContext->RunDir,
                pContext->RunNumStr, pContext->VersionNum);
  if (NULL == (SeedFile = fopen(SeedFileName,"w")))
  {
    SCRlog( "Master: Failed to open seed file." );
  }
  else
  {

    fprintf(SeedFile, "%s", temp_string);
    fclose(SeedFile);
  }


  SCRlog( "\tSeed = %ld\n", pContext->MasterSeed );
```

```
}

int32
Init( master_data_t *pContext )
{
 int32                  CLast;
 double                 CkptProximAddOffset;
 double                 CkptsPerInterval;
 char                   ElementString[32];
 FILE                   *Fd;
 char                   FileName[256];
 char                   IDString[7];
 char                   *List;
 int                    Index;
 int32                  RunId;
 int32                  Status;
 char                   *pTmpChar;
 struct timeval         TimeStamp;
 char                   TmpStr[256];



 SCRlog( "Master: Initializing.\n" );


 memset( pContext, 0, sizeof( *pContext ));



 pContext->pVersion = VERSION;
 SCRlog( "\tSoftware Version = %s\n", pContext->pVersion );
 PRTEset_network_variable("TPCC_SCRIPTS_VERSION", pContext-
>pVersion);


 pContext->OsPid = getpid();
 SCRlog( "\tOS PID = %d", pContext->OsPid );


 SCRlog( "\tMaster PRTE User ID = %d\n", User_id );


 Flush_log = TRUE;


 SCR_getnet_string( "TEST_RESULTS_DIR",
TEST_RESULTS_DIR_DEFAULT,
                    pContext->OutputDir );



 if( -1 == ( RunId = MakeRunDir( pContext->OutputDir, pContext->RunDir ))
 {
  return( SCRerr( ERR_MAKING_RUN_DIR ));
 }

 sprintf( TmpStr, "%d", RunId );
 PRTEset_network_variable( "RUN_ID", TmpStr );



 strcpy( Log_path, pContext->RunDir );
 strcpy( Log_name, "master.log" );



 Logging_type = ( SCRIPT_SPECIFIC_LOGGING );

 SCRlog( "\tLogFile = %s%s\n", Log_path, Log_name );


 PRTEcatch_message_binary( BinMsgHandler, pContext );
```

```
 sprintf( IDString, "%d", User_id );
 PRTEset_network_variable( "MASTER_ID", IDString );


 SCR_getnet_string( "RUN_NUMBER", RUN_NUMBER_DEFAULT,
                    pContext->RunNumStr );
 SCRlog( "\tRunNumber = %s.", pContext->RunNumStr );


 SCR_getnet_int( "VERSION_NUMBER", VERSION_NUMBER_DEFAULT,
                 &pContext->VersionNum );
 SCRlog( "\tVersionNumber = %d\n", pContext->VersionNum );


 SCR_getnet_double( "WARMUP_TIME", WARMUP_TIME_DEFAULT,
                    &pContext->WarmUpTime );
 if( 0 < pContext->WarmUpTime )
 {
  SCRlog( "\tWarmup = %7.2f", pContext->WarmUpTime );
 }

 SCR_getnet_double( "STEADY_STATE_TIME",
STEADY_STATE_TIME_DEFAULT,
                    &pContext->SteadyStateTime );
 if( 0 < pContext->SteadyStateTime )
 {
  SCRlog( "\tSteady State Time = %7.2f", pContext->SteadyStateTime );
 }

 SCR_getnet_double( "MEASUREMENT_INTERVAL",
MEASUREMENT_INTERVAL_DEFAULT,
                    &pContext->MeasurementInterval );
 if( 0 < pContext->MeasurementInterval )
 {
  SCRlog( "\tMeasurement Interval= %7.2f", pContext->MeasurementInterval );
 }

 SCR_getnet_double( "COOLDOWN_TIME",
COOLDOWN_TIME_DEFAULT,
                    &pContext->CoolDownTime );
 if( 0 < pContext->CoolDownTime )
 {
  SCRlog( "\tCooldown = %7.2f\n", pContext->CoolDownTime );
 }


 SCR_getnet_double( "MSG_TIMEOUT", MSG_TIMEOUT_DEFAULT,
                    &( pContext->MsgTimeout ));
 SCRlog( "\tMessage Timeout = %5.2f\n", pContext->MsgTimeout );


 GetSeed( pContext );



 SCR_getnet_int( "C_LAST", C_LAST_DEFAULT, &CLast );
 SCRlog( "\tC_LAST = %d\n", CLast );
 sprintf( FileName, "%srte%sclast.v%d", pContext->RunDir,
          pContext->RunNumStr, pContext->VersionNum );
 if( NULL == (Fd = fopen( FileName, "w" )))
 {
  ( void ) SCRerr( ERR_OPEN_C_LAST_FILE );
 }
 else
 {
  gettimeofday( &TimeStamp, NULL );
  fprintf( Fd, "Date: %s\n\n", ctime( &TimeStamp.tv_sec ));
  fprintf( Fd, "C_LAST constant = %d.\n", CLast );
```

```
  fclose( Fd );
 }


 SCR_getnet_int( "LOOPBACK_MODE", LOOPBACK_MODE_DEFAULT,
                              &pContext->LoopbackMode );
 switch( pContext->LoopbackMode )
 {
  case END_TO_END_MODE:
   SCRlog( "\tRunning in end-to-end mode.\n" );
   break;
  case BE_LOOPBACK_MODE:
   SCRlog( "\tRunning in back end loopback mode.\n" );
   break;
  case FE_LOOPBACK_MODE:
   SCRlog( "\tRunning in front end loopback mode.\n" );
   break;
  case RTE_LOOPBACK_MODE:
   SCRlog( "\tRunning in rte loopback mode.\n" );
   break;
  default:
   SCRlog( "LoopbackMode = %d", pContext->LoopbackMode );
   SCRerr( ERR_INVALID_LOOPBACK_MODE );
   break;
 }


 if( RTE_LOOPBACK_MODE != pContext->LoopbackMode )
 {


   SCR_getnet_string( "CGI_SCRIPT_NAME",
CGI_SCRIPT_NAME_DEFAULT,
                                  pContext->CgiScriptName );


   SCR_getnet_int( "ADMIN_USER_COUNT",
ADMIN_USER_COUNT_DEFAULT,
                              &pContext->AideCount );
   SCRlog( "\tAdmin User Count = %d\n", pContext->AideCount );

   if( 0 < pContext->AideCount )
   {

    SCR_getnet_int( "CHECKPOINT_INTERVAL",
CHECKPOINT_INTERVAL_DEFAULT,
                                  &pContext->CheckpointInterval );

    SCRlog( "\tCheckpoint Interval = %d", pContext->CheckpointInterval );


    if( 0 < pContext->CheckpointInterval )
    {
                SCR_getnet_double(
"CKPT_PROXIMITY_ADDITIONAL_OFFSET",

CKPT_PROXIMITY_ADDITIONAL_OFFSET_DEFAULT,
                                        &CkptProximAddOffset );




                if( pContext->CheckpointInterval > 1800 )
                {
                 return( SCRerr( ERR_CKPT_INT_TOO_LONG ));
                }


                if( pContext->CheckpointInterval > pContext-
>MeasurementInterval )
                {
                 return( SCRerr(
ERR_CKPT_INT_GREATER_THAN_MEASUREMENT_INT ));
                }



                if( (pContext->CheckpointInterval < pContext-
>MeasurementInterval) &&
                  ( (int32) pContext->MeasurementInterval %
                    pContext->CheckpointInterval != 0 ))
                {
                 return( SCRerr(
ERR_MEASUREMENT_NOT_INTEGRAL_MULTIPLE ));
                }



                CkptsPerInterval = ceil( pContext->MeasurementInterval /
                                                        pContext-
>CheckpointInterval );
                if( 4 > CkptsPerInterval )
                {
                 pContext->Proximity = (pContext->MeasurementInterval *
                                                        (2 / (CkptsPerInterval+3))) /
2;
                }



                if( 0 >= (pContext->CkptStartOffset = pContext->WarmUpTime +
                          pContext->Proximity + CkptProximAddOffset -
                          pContext->CheckpointInterval))
                {
                 return( SCRerr( ERR_WARMUP_TOO_SHORT ));
                }


                pContext->DoCheckpoints = TRUE;
                if( (0 < pContext->Proximity) && (0 < CkptProximAddOffset) )
                {
                 SCRlog( "\tProximity = %7.2f",
                          pContext->Proximity );
                 SCRlog( "\tAdditional Proximity Offset = %7.2f",
                          CkptProximAddOffset );
                 SCRlog( "\tTotal Checkpoint Offset = %7.2f\n",
                          pContext->Proximity + CkptProximAddOffset );
                } else if( 0 < pContext->Proximity )
                {
                 SCRlog( "\tProximity = %7.2f\n",
                          pContext->Proximity );
                }
                else if( 0 < CkptProximAddOffset )
                {
                 SCRlog( "\tCheckpoint Offset = %7.2f\n",
                          CkptProximAddOffset );
                }
    }
    else
    {
                SCRlog( "Master: Checkpoints will not be done." );
    }


    if( NULL == ( List = PRTEget_network_variable( "ADMIN_FE_NAMES"
)))
    {
                return( SCRerr( ERR_ADMIN_FE_NAMES ));
    }

    pContext->AdminFeCount = SCR_count_elements( List );
```

```
    if( pContext->AideCount != pContext->AdminFeCount )
    {

            return( SCRerr( ERR_ADMIN_USER_FE_COUNT_MISMATCH
));
    }



    if( NULL == ( pContext->pAides =
                        ( aide_user_t * ) malloc( pContext->AideCount *
                                                        sizeof(
aide_user_t ))))
    {
            return( SCRerr( ERR_MALLOC_ADMIN_USER_ARRAY ));
    }


    if( NULL == ( pContext->pAdminFEs =
                        ( admin_fe_t * ) malloc( pContext->AdminFeCount
*
                                                        sizeof(
admin_fe_t ))))
    {
            return( SCRerr( ERR_MALLOC_ADMIN_FE_ARRAY ));
    }


    Index = 0;
    pTmpChar = List;
    while( NULL !=
             ( pTmpChar = SCR_next_list_element( pTmpChar,
ElementString )))
    {
            pContext->pAdminFEs[Index].Name =
              ( char * ) malloc( sizeof( ElementString ));
            strcpy( pContext->pAdminFEs[Index].Name, ElementString );
            Index++;
    }
   }



  if(( END_TO_END_MODE == pContext->LoopbackMode ) &&
    ( 0 < pContext->AideCount ))
  {
    SCRlog( "\tAudit utilities will be run." );


    if( ERR_SUCCESS != ( Status = GetBeInfo( pContext )))
    {
            ( void ) SCRerr( ERR_GET_BE_INFO );
            return( Status );
    }

    if( ERR_SUCCESS != ( Status = GetDbType( &( pContext->DbType ))))
    {
            SCRlog( "Error getting the database type." );
            return( Status );
    }

    if( ERR_SUCCESS !=
              ( Status =
                UTILGetAuditFunctionNetworkVariables( &pContext-
>BeTasksData )))
    {
            SCRlog( "Error getting audit function net vars." );
            return( Status );
    }
  }
 }
 return( ERR_SUCCESS );
}


int32
MakeRunDir(char *DirName, char *RunDir)
{
#ifdef _WIN32
 HANDLE                 find_handle = INVALID_HANDLE_VALUE;
 WIN32_FIND_DATA  find_data;
#else
 DIR          *pDir;
 struct dirent *Entry;
#endif
 char         *c;
 char                   FormatStr[MAX_RUN_DIR_NAME_LEN + 3];
 int          MaxDir=0;
 int          NewDir=0;
 int          Len;
 mode_t       Mode = 0770;


 if( NULL == DirName )
  return -1;



#ifdef _WIN32
 strcpy(RunDir, DirName);
 strcat(RunDir,"/*");
 find_handle = FindFirstFile(( LPTSTR )RunDir, &find_data );
 if( INVALID_HANDLE_VALUE == find_handle )
  return -1;

 do
 {
  c = find_data.cFileName;
  while( (*c != '\0') && ('0' <= *c) && (*c <= '9') )
  {
   c++;
  }

  if ( '\0' == *c)
  {
   MaxDir = maximum(MaxDir, atoi(find_data.cFileName));
  }
 } while( FindNextFile( find_handle, &find_data ));

 (void)FindClose( find_handle );
#else
 if (NULL == (pDir = opendir(DirName)))
 {
  return -1;
 }


 for (Entry = readdir(pDir); Entry != NULL; Entry = readdir(pDir))
 {

  c = Entry->d_name;
  while( (c != '\0') && ('0' <= *c) && (*c <= '9') )
  {
   c++;
  }

  if ( '\0' == *c)
  {
   MaxDir = maximum(MaxDir, atoi(Entry->d_name));
  }
```

```
  }

  closedir(pDir);
#endif


  NewDir = MaxDir + 1;


  strcpy(RunDir,DirName);


  Len = strlen(RunDir);
  if ('/' != RunDir[Len])
  {
    RunDir[Len++] = '/';
  }



  sprintf(FormatStr,"%%0%dd/", MAX_RUN_DIR_NAME_LEN);
  sprintf(&RunDir[Len], FormatStr, NewDir);
  Len += 4;


  if ( -1 == MKDIR(RunDir, Mode) )
  {
    return -1;
  }
  SCRlog("\tRun Dir = %s", RunDir);
  PRTEset_network_variable("RUN_DIR", RunDir);




  strcat(RunDir, LOG_DIR);
  if (-1 == MKDIR(RunDir, Mode))
  {
    return -1;
  }
  PRTEset_network_variable("LOG_DIR", RunDir);
  RunDir[Len] = '\0';

  return NewDir;
}



void
TellAidesTo( master_data_t *pContext, int32 Task, int32 State )
{
  int32 BeIndex;
  generic_dotask_msg_t SendMsg;



  if (pContext->AideCount == 0)
    return;


  SCRlog( "\t%d aides being told to %s.", pContext->AideCount,
MsgNames[Task]);
  SendMsg.Type = Task;
  SendMsg.Data.State = State;

  pContext->RequestsSent = pContext->AideCount;
  pContext->ReplysReceived = 0;
  pContext->AidesFailed = FALSE;
  pContext->MsgTimedOut = TRUE;


  for(BeIndex = 0; BeIndex < pContext->AideCount; BeIndex++ )
  {
    SendMsg.Data.HostId = BeIndex + 1;
    strcpy( SendMsg.Data.SutName, pContext->pBEs[BeIndex].Name );
    PRTEmessage_to_user_binary( sizeof( SendMsg ), &SendMsg,
                                                    pContext-
>pAides[BeIndex].PrteID,
                                                    pContext-
>pAides[BeIndex].PrteID );
  }

  PRTEdelay( pContext->MsgTimeout );

  if( pContext->MsgTimedOut )
  {
    SCRlog( "\tTimed out waiting for %d aides to %s.",
                ( pContext->RequestsSent - pContext->ReplysReceived ),
                MsgNames[Task] );
    SCRlog( "\tPausing.\n\n" );
    PRTEpause();
  }
  else if( pContext->AidesFailed )
  {
    SCRlog( "\tAides failed to %s.", MsgNames[Task] );
    SCRlog( "\tPausing.\n\n" );
    PRTEpause();
  }
  else
  {
    SCRlog( "\t%d aides successfully did %s.\n", pContext->AideCount,
                MsgNames[Task]);
  }

  return;
}
```

tpcc_master.h

```
#ifndef TPCC_MASTER_H
#define TPCC_MASTER_H

#include <stdtypes.h>

#ifdef _WIN32
# define pid_t int
# define PATH_MAX MAX_PATH
#else
#endif


#define BEFORE 1
#define AFTER  2



typedef struct _fe_t
{
  char  *Name;
  int32 Ucnt;
} fe_t;


typedef struct _be_t
{
  char  *Name;
  double LogSize;
  double LogUsed;
} be_t;

typedef struct _aide_user_t
```

```
{
  int32         PrteID;
} aide_user_t;

typedef struct _admin_fe_t
{
  char *Name;
} admin_fe_t;

typedef struct _task_data_t {
  char          BeUname[32];
  char          DbUname[32];
  char          DbPassword[32];
  int32         DbType;
  boolean  GetAllAuditFiles;
  char          OraStatScriptPath[PATH_MAX];
  char          CustomBeforeTestScript[PATH_MAX];
  char          CustomAfterTestScript[PATH_MAX];
} task_data_t;



typedef struct _master_data_t
{
  int32                 AdminFeCount;
  admin_fe_t            *pAdminFEs;
  aide_user_t           *pAides;
  int32                 AideCount;
  int32                 RequestsSent;
  int32                 ReplysReceived;
  BOOL                  AidesFailed;
  BOOL                  MsgTimedOut;
  double        MsgTimeout;

  char                  CgiScriptName[128];
  BOOL                  DoCheckpoints;
  double        CkptStartOffset;
  int32                 LoopbackMode;
  double        Proximity;
  pid_t                 OsPid;
  char                  RunDir[PATH_MAX];
  int                   CheckpointInterval;
  double        WarmUpTime;
  double        SteadyStateTime;
  double        MeasurementInterval;
  double        CoolDownTime;
  int                   ReducerID;
  BOOL                  ReducerFailed;
  char                  OutputDir[PATH_MAX];
  char                  RunNumStr[8];
  int                   VersionNum;
  long                  MasterSeed;
  char                  *pVersion;
  int32                 NumBE;
  be_t                  *pBEs;

  char                  BeUname[32];
  char                  DbUname[32];
  char                  DbPassword[32];
  char                  DbName[32];
  int                   DbType;
  char                  OraStatScriptPath[PATH_MAX];
  char                  CustomBeforeTestScript[PATH_MAX];
  char                  CustomAfterTestScript[PATH_MAX];


  task_data_t   BeTasksData;
} master_data_t;

#endif
```

tpcc_user.c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/types.h>
#include <sys/stat.h>
#include <errno.h>
#ifdef _WIN32
# include <nt_lib.h>
# include <time.h>
#endif

#include <common.h>
#include <prte.h>
#include <msg.h>
#include <netvar.h>
#include <scr_util.h>
#include <stdtypes.h>
#include <tpcc_gen.h>
#include <tpcc_user.h>


#define HTTP_STATUS_OK 200



#define DELI_DATA_STR_1 "Previous Deliveries:<BR>"
#define DELI_DATA_STR_2 "<BR>"

#define MAX_STATUS_FIELD_WIDTH 10


#define MENU_BAR_PAGE           2
#define DY_FORM_PAGE            4
#define DY_RESP_PAGE            5
#define NO_FORM_PAGE            6
#define NO_RESP_PAGE            7
#define OS_FORM_PAGE            8
#define OS_RESP_PAGE            9
#define PT_FORM_PAGE            10
#define PT_RESP_PAGE            11
#define SL_FORM_PAGE            12
#define SL_RESP_PAGE            13


#define W_ID_STR "Warehouse: "
#define D_ID_STR "District: "
#define O_ID_STR "Order Number: "


#define DY_MENU_DELAY 0.1
#define NO_MENU_DELAY 0.1
#define OS_MENU_DELAY 0.1
#define PT_MENU_DELAY 0.1
#define SL_MENU_DELAY 0.1

#define DY_DISPLAY_DELAY 0.1
#define NO_DISPLAY_DELAY 0.1
#define OS_DISPLAY_DELAY 0.1
#define PT_DISPLAY_DELAY 0.1
#define SL_DISPLAY_DELAY 0.1

#define MENU_SYNC_STR HTTP_SYNC_STR
#define NO_SYNC_STR   HTTP_SYNC_STR
#define PT_SYNC_STR   HTTP_SYNC_STR
#define DY_SYNC_STR   HTTP_SYNC_STR
#define SL_SYNC_STR   HTTP_SYNC_STR
#define OS_SYNC_STR   HTTP_SYNC_STR
```

```
#define DY_MENU_STR \
"GET %s?3=d%d&0=Delivery HTTP/1.0\nConnection: Keep-Alive\n\n"
#define NO_MENU_STR \
"GET %s?3=n%d&0=NewOrder HTTP/1.0\nConnection: Keep-Alive\n\n"
#define OS_MENU_STR \
"GET %s?3=o%d&0=OrderStatus HTTP/1.0\nConnection: Keep-Alive\n\n"
#define PT_MENU_STR \
"GET %s?3=p%d&0=Payment HTTP/1.0\nConnection: Keep-Alive\n\n"
#define SL_MENU_STR \
"GET %s?3=s%d&0=StockLevel HTTP/1.0\nConnection: Keep-Alive\n\n"


#define SUBMIT "\n"


#define SET_WID_DID_STR \
"GET %s?3=W%d&4=%d&5=%d&0=Submit HTTP/1.0\nConnection: Keep-
Alive\n\n"



#define LOGOFF_STR \
"GET %s?3=M%d&0=Exit HTTP/1.0\nConnection: Keep-Alive\n\n"


#define NO_MENU_CNTX 06
#define PT_MENU_CNTX 14
#define OS_MENU_CNTX 22
#define DY_MENU_CNTX 30
#define SL_MENU_CNTX 38



#define MAXSTRINGSIZE   4096


#define TXN_WAIT_STATE_LOGIN 1
#define TXN_WAIT_STATE_RESUME 2
#define TXN_WAIT_STATE_STARVED 3
#define TXN_WAIT_STATE_RUNNING 4


int  Init(user_context_t *pContext);
int  ConnectSut();
int  ConnectDatabase(user_context_t *pContext);
void DisconnectExit(user_context_t *pContext, int status, int Disconnect);
int  TpccTxn(user_context_t *pContext, int type, double TxnKeying, double
TxnThink);



void TxnLogRoutine( void *txn_context, char *Data, int Bytes );
int  ParseWebPage( user_context_t *pContext, int ExpectedPageType );
int  ParseDyData(user_context_t *pContext);
int  ParseDura(user_context_t *pContext);
void BinMsgHandler(void *Context, int SendersID, int Len, void *Msg);


struct _txn
{
  char sync[8];
  int  menu_context;
  char menu_string[128];
  int  form_page_type;
  int  resp_page_type;
  double menu_delay;
  double txn_delay;
  void (*get_txn_data)();
};

struct _txn txn[6] = {
  { { '\0' }, 0, { '\0' }, 0, 0,
    0, 0, { '\0' }},
  { NO_SYNC_STR, NO_MENU_CNTX, NO_MENU_STR,
NO_FORM_PAGE, NO_RESP_PAGE,
    NO_MENU_DELAY, NO_DISPLAY_DELAY, &get_no_data},
  { PT_SYNC_STR, PT_MENU_CNTX, PT_MENU_STR, PT_FORM_PAGE,
PT_RESP_PAGE,
    PT_MENU_DELAY, PT_DISPLAY_DELAY, &get_pt_data},
  { OS_SYNC_STR, OS_MENU_CNTX, OS_MENU_STR, OS_FORM_PAGE,
OS_RESP_PAGE,
    OS_MENU_DELAY, OS_DISPLAY_DELAY, &get_os_data},
  { DY_SYNC_STR, DY_MENU_CNTX, DY_MENU_STR,
DY_FORM_PAGE, DY_RESP_PAGE,
    DY_MENU_DELAY, DY_DISPLAY_DELAY, &get_dy_data},
  { SL_SYNC_STR, SL_MENU_CNTX, SL_MENU_STR, SL_FORM_PAGE,
SL_RESP_PAGE,
    SL_MENU_DELAY, SL_DISPLAY_DELAY, &get_sl_data}
};



void
main(int argc, char *argv[])
{
  generic_status_msg_t   StatusMsg;
  user_context_t         Context;
  user_context_t         *pContext = &Context;
  int                            Txn;
  double                 TxnKeying;
  double                 TxnThink;


  PRTEinit(argc,argv);



  PRTEcatch_message_binary(BinMsgHandler, pContext);


  pContext->Txn[0] = -1;
  pContext->Txn[1] = -1;
  pContext->TxnKeying[0] = 0.0;
  pContext->TxnKeying[1] = 0.0;
  pContext->TxnThink[0] = 0.0;
  pContext->TxnThink[1] = 0.0;


  pContext->TxnWaitState = TXN_WAIT_STATE_LOGIN;
  PRTEpause();

  StatusMsg.Type = USER_LOGIN_REPLY_MSG;


  if( ERR_SUCCESS != ( StatusMsg.Status = Init( pContext )))
  {

    PRTEmessage_to_user_binary( sizeof( StatusMsg ), &StatusMsg,
                                                pContext->ReducerID,
pContext->ReducerID );
    DisconnectExit ( pContext, StatusMsg.Status, FALSE );
  }

  if( RTE_LOOPBACK_MODE != pContext->LoopbackMode )
  {

    if( ERR_SUCCESS != ( StatusMsg.Status = ConnectSut() ))
    {

      PRTEmessage_to_user_binary( sizeof( StatusMsg ), &StatusMsg,
                                                pContext->ReducerID,
pContext->ReducerID );
```

```
    DisconnectExit( pContext, StatusMsg.Status, FALSE );
   }

   if( ERR_SUCCESS != ( StatusMsg.Status = ConnectDatabase( pContext )))
   {

    PRTEmessage_to_user_binary( sizeof( StatusMsg ), &StatusMsg,
                                                    pContext->ReducerID,
pContext->ReducerID );
    DisconnectExit( pContext, StatusMsg.Status, TRUE );
   }
  }

  StatusMsg.Status = ERR_SUCCESS;
  PRTEmessage_to_user_binary( sizeof(StatusMsg), &StatusMsg,
                                        pContext->ReducerID, pContext-
>ReducerID );


  pContext->TxnWaitState = TXN_WAIT_STATE_RESUME;
  PRTEpause();

  StatusMsg.Type = USER_RESUME_REPLY_MSG;
  StatusMsg.Status = ERR_SUCCESS;
  PRTEmessage_to_user_binary( sizeof( StatusMsg ), &StatusMsg,
                                        pContext->ReducerID, pContext-
>ReducerID );


  while( !Run_time_complete && StatusMsg.Status == ERR_SUCCESS )
  {

   if( pContext->Txn[0] < NEW_ORDER_TXN || pContext->Txn[0] >
STOCK_LEVEL_TXN )
   {
    if( pContext->Txn[0] == -1 )
    {


            SCR_to_console_and_log(
                "Sleeping until a transaction message comes from reducer." );
            pContext->TxnWaitState = TXN_WAIT_STATE_STARVED;
            PRTEpause();
    }
    else
    {
            SCRlog( "Transaction type = %d", pContext->Txn[0] );
            StatusMsg.Status = SCRerr(
ERR_INVALID_TXN_TYPE_REQUESTED );
            break;
    }
   }




   Txn = pContext->Txn[0];
   pContext->Txn[0] = pContext->Txn[1];
   pContext->Txn[1] = -1;
   TxnKeying = pContext->TxnKeying[0];
   pContext->TxnKeying[0] = pContext->TxnKeying[1];
   pContext->TxnKeying[1] = -1.0;
   TxnThink = pContext->TxnThink[0];
   pContext->TxnThink[0] = pContext->TxnThink[1];
   pContext->TxnThink[1] = -1.0;


   StatusMsg.Status = TpccTxn( pContext, Txn, TxnKeying, TxnThink );
  }
```

```
  if( RTE_LOOPBACK_MODE == pContext->LoopbackMode )
  {
   DisconnectExit( pContext, StatusMsg.Status, FALSE );
  }
  else
  {
   DisconnectExit( pContext, StatusMsg.Status, TRUE );
  }
}


void
BinMsgHandler( void *Context, int SendersID, int Len, void *Msg )
{
  user_context_t *pContext;
  char      TempStr[256];




  generic_msg_t                                 *pMsg;
  txn_msg_t                         *pTxn;



  pMsg = Msg;
  pContext = Context;


  switch( pMsg->Type ) {
   case USER_STOP_REQUEST_MSG:
    PRTEstop( );
    PRTEdelay( 0.0 );
    break;

   case TXN_MSG:
    pTxn = Msg;
    if (pContext->Txn[0] == -1)
    {
            pContext->Txn[0] = pTxn->Data.Txn;
            pContext->TxnKeying[0] = pTxn->Data.TxnKeying;
            pContext->TxnThink[0] = pTxn->Data.TxnThink;
    }
    else if (pContext->Txn[1] == -1)
    {
            pContext->Txn[1] = pTxn->Data.Txn;
            pContext->TxnKeying[1] = pTxn->Data.TxnKeying;
            pContext->TxnThink[1] = pTxn->Data.TxnThink;
    }
    else
    {

            sprintf(TempStr, "We received one too many transactions from
Reducer.");
            SCR_to_console_and_log(TempStr);
    }

    if (TXN_WAIT_STATE_RUNNING != pContext->TxnWaitState)
    {

            if( TXN_WAIT_STATE_LOGIN == pContext->TxnWaitState )
            {

             pContext->ReducerID = SendersID;
            }
            if (TXN_WAIT_STATE_STARVED == pContext->TxnWaitState)
            {
             SCR_to_console_and_log("Waking from transaction message wait
sleep.");
```

```
                }
                PRTEresume_users(User_id, User_id);
                pContext->TxnWaitState = TXN_WAIT_STATE_RUNNING;
    }

    break;
   default:
    sprintf( TempStr, "User: Received unknown message type: %d", *pMsg);
    PRTEsend_console_message( TempStr );
  }
}

int
TpccTxn(user_context_t *pContext, int type, double TxnKeying, double
TxnThink)
{
  char context_string[8];
  char menu_buf[128];
  txn_timing_msg_t txn_timing;

  char TxnString[MAXSTRINGSIZE];
  int TxnContext;

  int Status;
  deferred_delivery_msg_t DyMsg;
  char DyMsgFiller[] = " 00000042 00001 02 0000 0001 0002 0003 0004 0005
0006 0007 0008 0009";
  char StartTime[11];



  sprintf(context_string,"%d",txn[type].menu_context);

  sprintf(menu_buf,txn[type].menu_string, pContext->cgi_script_name,
                WDID(pContext->w_id,pContext->d_id));


  pContext->WebPage.pBuf = pContext->WebPage.Buf;
  *pContext->WebPage.pBuf = '\0';

  if( RTE_LOOPBACK_MODE != pContext->LoopbackMode )
  {

    Status = PRTEtimed_send_and_sync(menu_buf, txn[type].sync,
context_string,

&txn_timing.Data.MenuStart,

&txn_timing.Data.MenuResponse);
    if (Status == -1)
    {
     SCRlog( "Sending/syncing for %s.", menu_buf );
     return E_WAIT;
    }


    Status = ParseWebPage( pContext, txn[type].form_page_type);
    if (Status != ERR_SUCCESS)
    {
     return Status;
    }
  }
  else
  {

    txn_timing.Data.MenuStart = PRTEtimer_begin(context_string);
    txn_timing.Data.MenuResponse = PRTEtimer_end();
  }

  if (Run_time_complete)
   return ERR_SUCCESS;

  (txn[type].get_txn_data)(pContext, TxnString, &TxnContext);

  sprintf(context_string, "%d", TxnContext);
  txn_timing.Data.TxnContext = TxnContext;


  PRTEdelay(TxnKeying + txn[type].menu_delay);

  if (Run_time_complete)
  {
   return ERR_SUCCESS;
  }


  pContext->WebPage.pBuf = pContext->WebPage.Buf;
  *pContext->WebPage.pBuf = '\0';

  if( RTE_LOOPBACK_MODE != pContext->LoopbackMode )
  {
    Status = PRTEtimed_send_and_sync(TxnString, txn[type].sync,
context_string,

&txn_timing.Data.TxnStart,

&txn_timing.Data.TxnResponse);
    if (Status == -1)
    {
     SCRlog( "Sending/syncing for %s.", TxnString );
     return E_WAIT;
    }


    Status = ParseWebPage(pContext, txn[type].resp_page_type);
    if (Status != ERR_SUCCESS)
    {
     SCR_to_console_and_log("Failed web request:");
     SCR_to_console_and_log(TxnString);
     return Status;
    }
  }
  else
  {

    txn_timing.Data.TxnStart = PRTEtimer_begin(context_string);
    txn_timing.Data.TxnResponse = PRTEtimer_end();

    if (INT_DELIVERY_TXN == type)
    {

     sprintf(StartTime, "%010d", time(NULL));
     memcpy(DyMsg.Data.DefDyRecord, StartTime, 10);
     memcpy(&(DyMsg.Data.DefDyRecord[10]), DyMsgFiller,
                DEF_DY_DATA_STR_SIZE - 10);
     DyMsg.Type = DEFERRED_DELIVERY_DATA_MSG;
     PRTEmessage_to_user_binary (sizeof(DyMsg), &DyMsg, pContext-
>ReducerID,
                                                pContext->ReducerID);
    }
  }

  if (Run_time_complete)
  {
   return ERR_SUCCESS;
  }


  if (pContext->ReducerID > 0)
  {
   txn_timing.Type = TXN_TIMING_MSG;
   PRTEmessage_to_user_binary (sizeof (txn_timing), &txn_timing,
```

```
                                                pContext-
>ReducerID,pContext->ReducerID);
  }

  if (!Run_time_complete)
  {

    PRTEdelay( TxnThink + txn[type].txn_delay);
  }


  return ERR_SUCCESS;
}



void
DisconnectExit(user_context_t *pContext, int Status, int Disconnect)
{
  user_exit_msg_t           ExitMsg;
  char tmp_buf[128];



  if ( TRUE == Disconnect )
  {
    sprintf(tmp_buf, LOGOFF_STR, pContext->cgi_script_name,
                    WDID(pContext->w_id,pContext->d_id));
    PRTEsend(tmp_buf);

    PRTEdisconnect_application();

  }



  Flush_log = TRUE;
  if (Status == ERR_SUCCESS)
  {
    sprintf(tmp_buf,"User %d: Exited gracefully as a swan.", User_id);
    PRTEto_log(110, tmp_buf);
  }
  else
  {
    SCRlog( "User %d: Exited with fatal error (%d).", User_id, Status );
  }


    ExitMsg.Type = USER_EXIT_MSG;
    ExitMsg.Data.Status = Status;
    ExitMsg.Data.ExitTime = PRTEtimer_begin ( "x" );
    PRTEtimer_cancel();
    PRTEmessage_to_user_binary (sizeof(ExitMsg), &ExitMsg,
                                                pContext->ReducerID,
pContext->ReducerID);

  PRTEexit();
}

int
ConnectSut(){
  double slept_time;
  double sleep_time;
  char sleep_string[120];
  int stat;
  int saved_flush_log;
  int ret_code;

  ret_code = ERR_SUCCESS;

  saved_flush_log = Flush_log;
  Flush_log = TRUE;

  sleep_time = 0.01;
  slept_time = 0;
  while ( -1 == (stat = PRTEconnect_application( "socket 80 %s", Sut_name ))
&&
                ( ECONNREFUSED == errno ))
  {
    if (( 1.0 < slept_time ) && ( 0 == (int)slept_time % 60 ))
    {
      sprintf(sleep_string, "Connection refused, slept %d minute(s).",
                    (int)slept_time / 60);
      SCR_to_console_and_log(sleep_string);
    }
    if (Run_time_complete)
    {
      sprintf(sleep_string,"User %d has been asked to stop. Exiting.",User_id);
      SCR_to_console_and_log(sleep_string);
      ret_code = -1;
    }
    else
    {
      PRTEdelay( sleep_time );
      slept_time += sleep_time;
      sleep_time *= 2;
    }
  }
  if (-1 == stat)
  {
    sprintf(sleep_string, "User %d has failed to log in, errno = %d",
                    User_id, errno);
    SCR_to_console_and_log(sleep_string);
    SCR_error(E_CONNECT, Sut_name);
    ret_code = E_CONNECT;
  }

  Flush_log = saved_flush_log;
  return ret_code;
}

int
ConnectDatabase(user_context_t *pContext)
{
  char str_buf[256];



  PRTEsut_log_routine( &(pContext->WebPage), SCR_webpage_callback);


  pContext->WebPage.pBuf = pContext->WebPage.Buf;
  *pContext->WebPage.pBuf = '\0';

  sprintf(str_buf, CONNECT_DB_STR, pContext->cgi_script_name);

  PRTEsync_string(HTTP_SYNC_STR);
  if (-1 == PRTEsend(str_buf))
  {
    SCR_error(E_SEND,str_buf);
    return E_SEND;
  }

  if (-1 == PRTEwait_for())
  {
    SCR_error(E_WAIT,str_buf);
    return E_WAIT;
  }


  if ( ERR_SUCCESS != ParseWebPage(pContext, GREETING_PAGE) )
  {
    sprintf(str_buf, "User %d: Failed to get the Greeting Form.", User_id);
```

```
  SCR_to_console_and_log(str_buf);
  return E_FAILURE;
 }

 sprintf(str_buf,SET_WID_DID_STR, pContext->cgi_script_name,
                 WDID(pContext->w_id,pContext->d_id),pContext->w_id,
pContext->d_id);


 pContext->WebPage.pBuf = pContext->WebPage.Buf;
 *pContext->WebPage.pBuf = '\0';

 PRTEsync_string(HTTP_SYNC_STR);
 if (-1 == PRTEsend(str_buf))
 {
  SCR_error(E_SEND,str_buf);
  return E_SEND;
 }

 if (-1 == PRTEwait_for())
 {
  SCR_error(E_WAIT,str_buf);
  return E_WAIT;
 }


 if( ERR_SUCCESS != ParseWebPage(pContext, MENU_BAR_PAGE) )
 {
  sprintf(str_buf, "Failed to get the Menu Bar Page.", User_id);
  SCR_to_console_and_log(str_buf);
  return E_FAILURE;
 }
 return(ERR_SUCCESS);
}


int
Init( user_context_t *pContext )
{
 char temp_string[128];
 char *pNetVarStr;
 char *master_seed;
 int  temp_int;


 pContext->WebPage.Inited = FALSE;
 pContext->DyOffsets.Inited = FALSE;




 SCR_getnet_string("LOG_DIR", LOG_DIR_DEFAULT, temp_string);
 if ('\0' == temp_string[0] )
 {
  return( SCRerr( ERR_LOG_DIR_NOT_SET ));
 }
 else
 {
  strcpy (Log_path, temp_string);
  strcpy (Err_path, temp_string);
 }


 sprintf(temp_string, "user_%d.err",User_id);
 strcpy (Err_name, temp_string);


 sprintf(temp_string, "user_%d.log",User_id);
 strcpy (Log_name, temp_string);


 SCR_getnet_int("TPCC_USER_LOG_TYPE",
TPCC_USER_LOG_TYPE_DEFAULT,
                        &Logging_type);
 Logging_type |= (USER_SUT_DATA_LOGGING);


 pContext->Version = VERSION;


 pNetVarStr = PRTEget_network_variable("TPCC_SCRIPTS_VERSION");
 if( NULL == pNetVarStr )
 {

  return( SCRerr( ERR_CODE_VERSION_NOT_SET ));
 }
 else if( STRING_MATCH != strcmp( pContext->Version, pNetVarStr ))
 {
  SCRlog( "User code version = %s, Master code version = %s.",
          pContext->Version, pNetVarStr );
  return( SCRerr( ERR_CODE_VERSION_MISMATCH ));
 }



 SCR_getnet_int("TPCC_USER_FLUSH_LOG",
TPCC_USER_FLUSH_LOG_DEFAULT,
                        &Flush_log);


 SCR_getnet_int( "LOOPBACK_MODE", LOOPBACK_MODE_DEFAULT,
                        &pContext->LoopbackMode );


 SCR_getnet_string("CGI_SCRIPT_NAME",
CGI_SCRIPT_NAME_DEFAULT,
                        pContext->cgi_script_name);
 if( '\0' == *pContext->cgi_script_name )
 {
  return( SCRerr( ERR_CGI_SCRIPT_NAME_NOT_SET ));
 }



 master_seed = PRTEwait_for_network_variable("SEED");
 pContext->myseed = atol(master_seed) * User_id;
 if ((pContext->myseed % 2) == 0)
 {

  pContext->myseed--;
 }

 PRTErandomize(((unsigned) pContext->myseed % 100));
 srand48(pContext->myseed);


 SCR_getnet_bool("DURABILITY_LOGGING",
DURABILITY_LOGGING_DEFAULT,
                        &pContext->DurabilityLogging);


 temp_int = User_id - 1;
 pContext->w_id = (int) temp_int / 10;
 pContext->w_id += 1;

 pContext->d_id = temp_int % 10;
 pContext->d_id += 1;


 if( NULL == (pNetVarStr = PRTEget_network_variable( "MAX_W_ID" )))
 {
  return( SCRerr( ERR_MAX_W_ID_NOT_SET ));
```

```
  }
  pContext->w_max = atoi( pNetVarStr );


  SCR_getnet_int( "C_LAST", C_LAST_DEFAULT, &( pContext->CLast ));

  return( ERR_SUCCESS );
}



int
ParseWebPage(user_context_t *pContext, int ExpectedPageType)
{
  int ReturnStatus;
  char *pTmp;
  char TmpStr[256];
  int FormID;
  static char FormIDList[] = FORM_ID_INITIALIZER;


  if (!pContext->WebPage.Inited)
  {

if (NULL == (pTmp = strstr (pContext->WebPage.Buf, HTTP_STATUS_STR)))
    {

      sprintf(TmpStr, "User %d: Couldn't determine HTTP status offset.",
                    User_id);
      SCR_to_console_and_log(TmpStr);


      SCR_to_console_and_log("     Don't know how to parse this page...");
      SCR_to_console_and_log(pContext->WebPage.Buf);
      return(E_FAILURE);
    }
    else
    {
      pContext->WebPage.HttpStatusOffset = (pTmp - pContext->WebPage.Buf) +
                strlen(HTTP_STATUS_STR);
    }


    if (NULL == (pTmp = strstr (pTmp, FORM_STR)))
    {


      sprintf(TmpStr, "User %d: Failed to locate form offset.", User_id);
      SCR_to_console_and_log(TmpStr);
      SCR_to_console_and_log(pContext->WebPage.Buf);


      return(atoi(&pContext->WebPage.Buf[pContext-
>WebPage.HttpStatusOffset]));
    }
    else
    {
      pContext->WebPage.FormStrOffset = pTmp - pContext->WebPage.Buf;
      pContext->WebPage.FormStrLen = strlen(FORM_STR);
      pContext->WebPage.FormValOffset = pContext->WebPage.FormStrOffset +
                pContext->WebPage.FormStrLen;
      pContext->WebPage.ReqMinLen = pContext->WebPage.FormValOffset +
FORM_ID_LEN;
    }


    pContext->WebPage.Inited = TRUE;
  }


  if ((pContext->WebPage.pBuf - pContext->WebPage.Buf) < pContext-
>WebPage.ReqMinLen)
  {

      sprintf(TmpStr, "User %d: Recieved a web page not from our dll.",
                    User_id);
      SCR_to_console_and_log(TmpStr);
      SCR_to_console_and_log(pContext->WebPage.Buf);


      return(atoi(&pContext->WebPage.Buf[pContext-
>WebPage.HttpStatusOffset]));
  }
  else
  {


    if (0 != strncmp(pContext->WebPage.Buf + pContext-
>WebPage.FormStrOffset,
                                FORM_STR, pContext->WebPage.FormStrLen))
    {

      sprintf(TmpStr, "User %d: Recieved a web page without our form tag.",
                    User_id);
      SCR_to_console_and_log(TmpStr);
      SCR_to_console_and_log(pContext->WebPage.Buf);


      return(atoi(&pContext->WebPage.Buf[pContext-
>WebPage.HttpStatusOffset]));
    }


    FormID = *(pContext->WebPage.Buf + pContext->WebPage.FormValOffset);
    if (FormIDList[ExpectedPageType] != FormID)
    {

      if (FormIDList[ERROR_PAGE] == FormID)
      {
                if (2 != sscanf(pContext->WebPage.Buf,
"%s"ERROR_VALUE_STR,
                                        TmpStr, &ReturnStatus))
                {

                  sprintf(TmpStr, "User %d: Unable to parse error page from our
dll.",
                                User_id);
                  SCR_to_console_and_log(TmpStr);
                  SCR_to_console_and_log(pContext->WebPage.Buf);
                  return(atoi(&pContext->WebPage.Buf[pContext-
>WebPage.HttpStatusOffset]));
                }
                else {

                  sprintf(TmpStr, "User %d: Recieved error page (%d) from our
dll.",
                                User_id, ReturnStatus);
                  SCR_to_console_and_log(TmpStr);
                  SCR_to_console_and_log(pContext->WebPage.Buf);
                  return(ReturnStatus);
                }
      }
      else
      {

                sprintf(TmpStr, "User %d: Expected form ID: %c\nBut received:\n",
                        User_id, FormID);
                SCR_to_console_and_log(TmpStr);
                SCR_to_console_and_log(pContext->WebPage.Buf);
                return(E_FAILURE);
      }
```

```
  }
  else
  {

   ReturnStatus = ERR_SUCCESS;


   switch(ExpectedPageType)
   {
            case DY_RESP_PAGE:
             if (SUCCESS != ParseDyData(pContext))
             {
              sprintf(TmpStr, "User %d: Unable to parse deferred deli data.",
                        User_id);
              SCR_to_console_and_log(TmpStr);
             }
             break;

            case MENU_BAR_PAGE:
             break;

            case NO_RESP_PAGE:
             if(pContext->DurabilityLogging)
             {
              if (SUCCESS != ParseDura(pContext))
              {
               sprintf(TmpStr, "User %d: Durability parsing error.", User_id);
               SCR_to_console_and_log(TmpStr);
              }
             }
             break;

            default:

             break;
   }
  }
  return(ReturnStatus);
 }
}

int
ParseDyData(user_context_t *pContext)
{
 char *DyOffsetStr[] =
 {
  "Previous Deliveries:<BR>",
  "<BR>"
 };

 int32 Offset;
 int32 StartTime;
 char *pTmp;
 char TmpStr[256];
 deferred_delivery_msg_t DyMsg;


 if (!pContext->DyOffsets.Inited)
 {
  pTmp = pContext->WebPage.Buf;
  for (Offset = 0; Offset < DEFERRED_DY_RECORDS_PER_PAGE; Offset++)
  {
   if (NULL == (pTmp = strstr(pTmp, DyOffsetStr[Offset])))
   {
            sprintf(TmpStr, "User %d: Unable to get offset for deli record %d.",
                      User_id, Offset);
            SCR_to_console_and_log(TmpStr);
            return(E_FAILURE);
   }
   else
   {
            pContext->DyOffsets.Pos[Offset] = (pTmp - pContext->WebPage.Buf) +
              strlen(DyOffsetStr[Offset]);
            pTmp += strlen(DyOffsetStr[Offset]);
   }
  }
 }


 for (Offset = 0; Offset < DEFERRED_DY_RECORDS_PER_PAGE; Offset++)
 {
  pTmp = pContext->WebPage.Buf + pContext->DyOffsets.Pos[Offset];
  StartTime = 0;


  while ('0' <= *pTmp && *pTmp <= '9')
  {
   StartTime = (StartTime * 10) + (*pTmp - '0');
   pTmp++;
  }


  if (0 != StartTime)
  {
   DyMsg.Type = DEFERRED_DELIVERY_DATA_MSG;
   strncpy(DyMsg.Data.DefDyRecord,
            pContext->WebPage.Buf + pContext->DyOffsets.Pos[Offset],
            DEF_DY_DATA_STR_SIZE);
   PRTEmessage_to_user_binary (sizeof(DyMsg), &DyMsg, pContext->ReducerID,
                                                    pContext->ReducerID);
  }
 }
 return(SUCCESS);
}


int
ParseDura(user_context_t *pContext)
{
 char *pTmp;
 char TmpStr[256];
 durability_data_msg_t DuraMsg;



 if( NULL == strstr( pContext->WebPage.Buf,
                        "Execution Status: Transaction commited." ))
 {



  return SUCCESS;
 }

 if( NULL == (pTmp = strstr(pContext->WebPage.Buf, W_ID_STR)))
 {

  sprintf(TmpStr, "User %d: Durability parsing error (warehouse).", User_id);
  SCR_to_console_and_log(TmpStr);
  return FAILURE;
 }
 pTmp += strlen( W_ID_STR );
 DuraMsg.Data.WarehouseId = atoi( pTmp );

 if( NULL == (pTmp = strstr(pTmp, D_ID_STR)))
 {

  sprintf(TmpStr, "User %d: Durability parsing error (district).", User_id);
  SCR_to_console_and_log(TmpStr);
  return FAILURE;
 }
```

```
  pTmp += strlen( D_ID_STR );
  DuraMsg.Data.DistrictId = atoi( pTmp );

  if( NULL == (pTmp = strstr(pTmp, O_ID_STR)))
  {

    sprintf(TmpStr, "User %d: Durability parsing error (order id).", User_id);
    SCR_to_console_and_log(TmpStr);
    return FAILURE;
  }
  pTmp += strlen( O_ID_STR );
  DuraMsg.Data.OrderId = atoi( pTmp );

  DuraMsg.Type = DURABILITY_DATA_MSG;
  PRTEmessage_to_user_binary (sizeof(DuraMsg), &DuraMsg, pContext-
>ReducerID,
                                          pContext->ReducerID);

  return(SUCCESS);
}
```

tpcc_user.h

```
#ifndef TPCC_USER_H
#define TPCC_USER_H


#include <scr_util.h>

/*+
17-Jan-97    WCarr                    Added new header format.  Modified code
to
                                      name cgi script (dll)
dynamically.  Moved
                                      transaction type definitions to
this file
                                      to be available to the run-time
reduction
                                      script.
 *
 * Copyright 1997 (c) Compaq Computer Corporation
 *
 * Description:
 *              This file defines values needed to run tpcc user scripts.
 *
-*/


#define DEFERRED_DY_RECORDS_PER_PAGE 2



struct _dy_offsets_t {
  BOOL Inited;
  int  Pos[DEFERRED_DY_RECORDS_PER_PAGE];
};
typedef struct _dy_offsets_t dy_offsets_t;




typedef struct _user_context_t {
  int32         CLast;
  int32 LoopbackMode;
  BOOL       DurabilityLogging;
  int w_id;
  int d_id;
  int w_max;
  int TxnWaitState;
  int32 Txn[2];
  double TxnThink[2];
  double TxnKeying[2];
  char cgi_script_name[128];
  char db_server_name[64];
  int ReducerID;
  long myseed;
  web_page_t WebPage;
  dy_offsets_t DyOffsets;
  char *Version;
} user_context_t;

#endif
```

Appendix D

Digital UNIX Tunable Parameters

Back-End Server

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

/etc/sysconfigtab

########################################


#####Copyright (c) Digital Equipment Corporation, 1991, 1997    *
# *                                                             *
# *   All Rights Reserved.  Unpublished rights  reserved  under  *
# *   the copyright laws of the United States.                  *
# *                                                             *
# *   The software contained on this media  is  proprietary  to  *
# *   and  embodies  the  confidential  technology  of  Digital  *
# *   Equipment Corporation.  Possession, use,  duplication  or  *
# *   dissemination of the software and media is authorized only  *
# *   pursuant to a valid written license from Digital Equipment  *
# *   Corporation.                                              *
# *                                                             *
# *   RESTRICTED RIGHTS LEGEND   Use, duplication, or disclosure  *
# *   by the U.S. Government is subject to restrictions  as  set  *
# *   forth in Subparagraph (c)(1)(ii)  of  DFARS  252.227-7013,  *
# *   or  in  FAR 52.227-19, as applicable.                     *
# *                                                             *
#
*****************************************************************
*
#
# HISTORY
#
# (c) Copyright 1990, 1991, 1992, 1993 OPEN SOFTWARE FOUNDATION, INC.
# ALL RIGHTS RESERVED
#
#
# OSF/1 1.2

ipc:
        shm-max = 2147483647
        ssm-threshold = 0

io:

generic:
        msgbuf_size = 12288

proc:
        max-per-proc-address-space = 163139607552
        max-per-proc-data-size = 163139607552
        max-per-proc-stack-size = 163139607552
        max-proc-per-user = 1024
        per-proc-address-space = 163139607552
        per-proc-data-size = 163139607552
        per-proc-stack-size = 163139607552
        sched-min-idle = 10

vfs:
        bufpages = 4000

vm:
        contig-malloc-percent = 1
        dump-user-pte-pages = 1
        gh-chunks = 4000
        new-wire-method = 0
        private-text = 1
        vm-maxvas = 163139607552
        vm-syswiredpercent = 95
        vm_mapentries = 4000
        vm_pagemax = 262144

rt:
        aio-max-num = 12288
        aio-max-retry = 2
        aio-task-max-num = 1024

net:
        netisrthreads = 1

pcount:
        Subsystem_Description = pcount device driver
        Module_Config_Name = pcount
        Module_Type = Dynamic
#       Device_Major_Req = Same
        Device_Char_Major = ANY
        Device_Char_Minor = 0
        Device_Char_Files = pcount0



/usr/sys/conf/CLIPPER

#####################################

ident           "CLIPPER"

options         UERF
options         OSF
options         _LMF_
options         BIN_COMPAT
options         COMPAT_43
options         MACH
options         MACH_IPC_TCACHE
options         MACH_IPC_WWA
options         MACH_IPC_XXXHACK
options         BUFCACHE_STATS
options         INOCACHE_STATS
options         STAT_TIME
options         VAGUE_STATS
options         UFS
options         NFS
options         NFS_SERVER
options         STRKINFO
options         STREAMS
options         LDTTY
options         RPTY
options         INET
options         UIPC
options         SYSV_COFF
options         QUOTA
options         LABELS
options SL
options SNMPINFO
options         DLI
options         BSD_TTY
options BPARM
options         FFM_FS
options PROCFS

#
# Standard options.
#
options         UNIX_LOCKS
options         SER_COMPAT
options         RT_PREEMPT
options         RT_SCHED
options         RT_SCHED_RQ
options         RT_PML
options         RT_TIMER
options         RT_SEM
options         RT_CSEM
options         RT_IPC
#
makeoptions     LOADADDR="ffffffc0000430000"
```

```
makeoptions     CDEBUGOPTS="-g3"
makeoptions     CCOMPRESS="-compress"
makeoptions     PROFOPTS="-DPROFILING -DPROFTYPE=4"

#
# Max number of processors in the system (DO NOT CHANGE)
#
processors      16

#
# Special options (see configuring the kernel chapter
# in the Guide to System Administration)
#
dfldsiz         163139607552
maxdsiz         163139607552
dflssiz         163139607552
maxssiz         163139607552
cpu             "DEC6600"
maxusers        32
config          vmunix  swap generic

bus             pci1       at nexus?    slot  1
callout after_c "../bin/mkdata pci"
bus             pci2001    at pci1      slot  1
bus             i2o0       at pci2001   slot  4
controller      i2o_bs0    at i2o0      slot  0
device disk     ri0        at i2o_bs0   drive     0
bus             pci3001    at pci1      slot  2
bus             i2o1       at pci3001   slot  4
controller      i2o_bs1    at i2o1      slot  0
device disk     ri128      at i2o_bs1   drive     128
bus             pci4001    at pci1      slot  3
bus             i2o2       at pci4001   slot  4
controller      i2o_bs2    at i2o2      slot  0
device disk     ri256      at i2o_bs2   drive     256
bus             pci5001    at pci1      slot  4
bus             i2o3       at pci5001   slot  4
controller      i2o_bs3    at i2o3      slot  0
device disk     ri384      at i2o_bs3   drive     384
bus             pci6001    at pci1      slot  5
bus             i2o4       at pci6001   slot  4
controller      i2o_bs4    at i2o4      slot  0
device disk     ri513      at i2o_bs4   drive     513
bus             pci7001    at pci1      slot  6
bus             itpsa0     at pci7001   slot  0 vector    itpsaintr
controller      scsi0      at itpsa0    slot  0
bus             itpsa1     at pci7001   slot  1 vector    itpsaintr
controller      scsi1      at itpsa1    slot  0
controller      tu0        at pci7001   slot  2
bus             pci0       at nexus?    slot  0
bus             pci2000    at pci0      slot  1
bus             i2o5       at pci2000   slot  4
controller      i2o_bs5    at i2o5      slot  0
device disk     ri640      at i2o_bs5   drive     640
bus             pci3000    at pci0      slot  2
bus             i2o6       at pci3000   slot  4
controller      i2o_bs6    at i2o6      slot  0
device disk     ri768      at i2o_bs6   drive     768
bus             pci4000    at pci0      slot  3
bus             i2o7       at pci4000   slot  4
controller      i2o_bs7    at i2o7      slot  0
device disk     ri896      at i2o_bs7   drive     896
bus             pci5000    at pci0      slot  4
bus             i2o8       at pci5000   slot  4
controller      i2o_bs8    at i2o8      slot  0
device disk     ri1024     at i2o_bs8   drive     1024
bus             isa0       at pci0      slot  7
callout after_c "../bin/mkdata isa"
controller      gpc0       at isa0      slot  0 vector    gpcintr
controller      ace0       at isa0      slot  2 vector    aceintr
controller      ace1       at isa0      slot  3 vector    aceintr
controller      lp0        at isa0      slot  4 vector    lpintr
controller      fdi0       at isa0      slot  5
device disk     fd0        at fdi0      drive     0
bus             ata0       at pci0      slot  15
controller      scsi2      at ata0      slot  0
controller      scsi3      at ata0      slot  1
bus             usb0       at pci0      slot  19

#
# Static Driver Definitions
#

#
# Pseudodevice Definitions (see configuring the
# kernel chapter in the Guide to System Administration)
#
pseudo-device   sysv_hab
pseudo-device   svid_three_hab
pseudo-device   svr_four_hab
pseudo-device   soe_two_hab
pseudo-device   rt_hab
pseudo-device   ether
pseudo-device   loop
pseudo-device   lv      2
pseudo-device   ws
```

Sybase Tunable Parameters

```
############################################################
##############
#
#                       Configuration File for the Sybase SQL Server
#
#                       Please read the System Administration Guide (SAG)
#                       before changing any of the values in this file.
#
############################################################
##############



[Configuration Options]

[General Information]

[Backup/Recovery]
        recovery interval in minutes = 32767
        print recovery information = DEFAULT
        tape retention in days = DEFAULT

[Cache Manager]
        number of oam trips = DEFAULT
        number of index trips = DEFAULT
        procedure cache percent = 1
        memory alignment boundary = DEFAULT
        global async prefetch limit = 0
        global cache partition number  = DEFAULT

[Named Cache:c_customer]
        cache size = 40M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[2K I/O Buffer Pool]
        pool size = 40M
        wash size = 4096 K
        local async prefetch limit = 0

[Named Cache:c_customer_index]
        cache size = 1600M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[2K I/O Buffer Pool]
        pool size = 1600M
        wash size = 307200 K
        local async prefetch limit = 0

[Named Cache:c_log]
```

```
        cache size = 115M
        cache status = log only
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 1

    [2K I/O Buffer Pool]
        pool size = 10M
        wash size = 512 K
        local async prefetch limit = 0

    [8K I/O Buffer Pool]
        pool size = 105M
        wash size = 8192 K
        local async prefetch limit = 0

    [Named Cache:c_no_ol]
        cache size = 700M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

    [2K I/O Buffer Pool]
        pool size = 700M
        wash size = 1024 K
        local async prefetch limit = 0

    [Named Cache:c_ol_index]
        cache size = 600M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

    [2K I/O Buffer Pool]
        pool size = 600M
        wash size = 4096 K
        local async prefetch limit = 0

    [Named Cache:c_orders]
        cache size = 900M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

    [2K I/O Buffer Pool]
        pool size = 675M
        wash size = 66560 K
        local async prefetch limit = 0

    [16K I/O Buffer Pool]
        pool size = 225M
        wash size = 20480 K
        local async prefetch limit = 0

    [Named Cache:c_stock]
        cache size = 9630M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

    [2K I/O Buffer Pool]
        pool size = 9630M
        wash size = 5527712 K
        local async prefetch limit = 0

    [Named Cache:c_stock_index]
        cache size = 470M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

    [2K I/O Buffer Pool]
        pool size = 470M
        wash size = 1024 K
        local async prefetch limit = 0

    [Named Cache:c_sysindex]
        cache size = 5M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 1

    [2K I/O Buffer Pool]
        pool size = 5M
        wash size = 256 K
        local async prefetch limit = 0

    [Named Cache:c_tinyhot]
        cache size = 100M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 1

    [2K I/O Buffer Pool]
        pool size = 90M
        wash size = 1024 K
        local async prefetch limit = 0

    [16K I/O Buffer Pool]
        pool size = 10M
        wash size = 256 K
        local async prefetch limit = 0

    [Named Cache:default data cache]
        cache size = 100M
        cache status = default data cache
        cache replacement policy = DEFAULT
        local cache partition number = 1

    [2K I/O Buffer Pool]
        pool size = 80M
        wash size = 1024 K
        local async prefetch limit = 0

    [16K I/O Buffer Pool]
        pool size = 10M
        wash size = 1024 K
        local async prefetch limit = 0

    [Meta-Data Caches]
        number of open databases = DEFAULT
        number of open objects = DEFAULT
        open object spinlock ratio = DEFAULT
        number of open indexes = DEFAULT
        open index hash spinlock ratio = DEFAULT
        open index spinlock ratio = DEFAULT

    [Disk I/O]
        allow sql server async i/o = DEFAULT
        disk i/o structures = DEFAULT
        page utilization percent = DEFAULT
        number of devices = 83
        disable disk mirroring = DEFAULT
        disable character set conversions = DEFAULT
        enable unicode conversions = DEFAULT
        size of unilib cache = DEFAULT

    [Network Communication]
        default network packet size = DEFAULT
        max network packet size = 4096
        remote server pre-read packets = DEFAULT
        number of remote connections = DEFAULT
        allow remote access = DEFAULT
        number of remote logins = DEFAULT
        number of remote sites = DEFAULT
        max number network listeners = DEFAULT
        tcp no delay = DEFAULT
        allow sendmsg = DEFAULT
        syb_sendmsg port number = DEFAULT

    [O/S Resources]
        max async i/os per engine = 778
        max async i/os per server = 778

    [Parallel Query]
        number of worker processes = DEFAULT
        memory per worker process = DEFAULT
        max parallel degree = DEFAULT
        max scan parallel degree = DEFAULT
```

```
[Physical Resources]

[Physical Memory]
        total memory = 8192000
        additional network memory = 4915200
        lock shared memory = DEFAULT
        shared memory starting address = DEFAULT
        max SQL text monitored = DEFAULT

[Processors]
        max online engines = 4
        min online engines = DEFAULT

[SQL Server Administration]
        default database size = DEFAULT
        identity burning set factor = DEFAULT
        allow nested triggers = DEFAULT
        allow updates to system tables = DEFAULT
        print deadlock information = DEFAULT
        default fill factor percent = DEFAULT
        default exp_row_size percent = DEFAULT
        number of mailboxes = DEFAULT
        number of messages = DEFAULT
        number of alarms = DEFAULT
        number of pre-allocated extents = DEFAULT
        event buffers per engine = DEFAULT
        cpu accounting flush interval = DEFAULT
        i/o accounting flush interval = DEFAULT
        sql server clock tick length = DEFAULT
        runnable process search count = DEFAULT
        i/o polling process count = DEFAULT
        time slice = DEFAULT
        deadlock retries = DEFAULT
        cpu grace time = DEFAULT
        number of sort buffers = 0
        number of large i/o buffers = DEFAULT
        size of auto identity column = DEFAULT
        identity grab size = DEFAULT
        page lock promotion HWM = DEFAULT
        page lock promotion LWM = DEFAULT
        page lock promotion PCT = DEFAULT
        housekeeper free write percent = 0
        enable housekeeper GC = DEFAULT
        partition groups = DEFAULT
        partition spinlock ratio = DEFAULT
        allow resource limits = DEFAULT
        number of aux scan descriptors = DEFAULT
        SQL Perfmon Integration = DEFAULT
        allow backward scans = DEFAULT
        row lock promotion HWM = DEFAULT
        row lock promotion LWM = DEFAULT
        row lock promotion PCT = DEFAULT
        license information = DEFAULT

[User Environment]
        number of user connections = 300
        stack size = DEFAULT
        stack guard size = DEFAULT
        permission cache entries = DEFAULT
        user log cache size = 4096
        user log cache spinlock ratio = 10

[Lock Manager]
        number of locks = 40000
        deadlock checking period = DEFAULT
        freelock transfer block size = DEFAULT
        max engine freelocks = 50
        lock spinlock ratio = DEFAULT
        lock hashtable size = DEFAULT
        lock scheme = DEFAULT
        lock wait period = DEFAULT
        read committed with lock = DEFAULT

[Security Related]
        systemwide password expiration = DEFAULT
        audit queue size = DEFAULT
        curread change w/ open cursors = DEFAULT
        allow procedure grouping = DEFAULT
        select on syscomments.text = DEFAULT
        auditing = DEFAULT
        current audit table = DEFAULT
        suspend audit when device full = DEFAULT
        max roles enabled per user = DEFAULT
        unified login required = DEFAULT
        use security services = DEFAULT
        msg confidentiality reqd = DEFAULT
        msg integrity reqd = DEFAULT
        msg replay detection reqd = DEFAULT
        msg origin checks reqd = DEFAULT
        msg out-of-seq checks reqd = DEFAULT
        secure default login = DEFAULT
        dump on conditions = DEFAULT

[Extended Stored Procedure]
        esp unload dll = DEFAULT
        esp execution priority = DEFAULT
        esp execution stacksize = DEFAULT
        xp_cmdshell context = DEFAULT
        start mail session = DEFAULT

[Error Log]
        event logging = DEFAULT
        log audit logon success = DEFAULT
        log audit logon failure = DEFAULT
        event log computer name = DEFAULT

[Rep Agent Thread Administration]
        enable rep agent threads = DEFAULT
        maximum dump conditions = DEFAULT

[Component Integration Services]
        enable cis = DEFAULT
        cis connect timeout = DEFAULT
        cis bulk insert batch size = DEFAULT
        max cis remote connections = DEFAULT
        max cis remote servers = DEFAULT
        cis packet size = DEFAULT
        cis cursor rows = DEFAULT
        cis rpc handling = DEFAULT
```

init_server.sh

```
#!/bin/sh -f


isql -Usa -P <<- EOF

use tpcc
go

dbcc tune(maxwritedes, 10)
go

dbcc tune("doneinproc", 0)
go

dbcc tune("cleanup", 0)
go


set rowcount 1
go

select * from stock
go

select * from customer
go

select * from order_line
go

select * from orders
go
```

```
select * from new_order
go

set rowcount 0
go

dbcc iosize(tpcc, district, 16)
go

dbcc iosize(tpcc, warehouse, 16)
go

dbcc iosize(tpcc, item, 16)
go

dbcc iosize(tpcc, history, 16)
go

exec sp_logiosize "8"
go

select count(*) from warehouse(index warehouse prefetch 16 lru)
go

select count(*) from district(index district prefetch 16 lru)
go

select count(*) from item(index item prefetch 16 lru)
go

set rowcount 1
go

select * from history
go

dbcc tune(elcsize,128)
go

dbcc tune(des_bind, 5, warehouse)
dbcc tune(des_bind, 5, district)
dbcc tune(des_bind, 5, item)
dbcc tune(des_bind, 5, stock)
dbcc tune(des_bind, 5, order_line)
dbcc tune(des_bind, 5, orders)
dbcc tune(des_bind, 5, new_order)
dbcc tune(des_bind, 5, customer)
dbcc tune(des_bind, 5, history)

dbcc tune(des_bind, 5, neworder_local)
dbcc tune(des_bind, 5, neworder_remote)
dbcc tune(des_bind, 5, payment_byid)
dbcc tune(des_bind, 5, payment_byname)
dbcc tune(des_bind, 5, order_status_byid)
dbcc tune(des_bind, 5, order_status_byname)
dbcc tune(des_bind, 5, delivery)
dbcc tune(des_bind, 5, stock_level)

go


dbcc traceoff(-1)
go
```

Appendix E

Auditor Attestation

Benchmark Sponsor:

Dave Stanley
Compaq Computer
110 Spit Brook Road
ZKO2-3/M31
Nashua NH, 03062

February 2, 2000

I verified the TPC Benchmark™ C performance of the following Client Server configuration:

Platform:	**Compaq AlphaServer ES40 6/667 4 CPU c/s**
Operating system:	**Tru64 UNIX V4.0G**
Database Manager:	**Sybase Adaptive Server Enterprise 11.9.3**
Transaction Manager:	**Application Optimizer WNT Server Edition**

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	**tpmC**
4 x Alphachip 21264 (667 MHz)	16 GB (8 mb cache per processor)	183 x 9.1 GB	1.538 Seconds	**30,738.97**
Four (4) Clients: Compaq Proliant 1600 (Specification for each)				
1 x Pentium III (600 MHz)	640 MB	1 x 18 GB	n/a	n/a

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

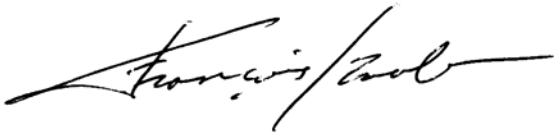
- The database records were the proper size
- The database was properly scaled and populated
- The required ACID properties were met

- The transactions were correctly implemented
- 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.
- All 90% response times were under the specified maximums
- At least 90% of all delivery transactions met the 80 Second completion time limit
- The reported measurement interval was 120 minutes
- The reported measurement interval was representative of steady state conditions
- Four checkpoints were taken during the reported measurement interval
- The repeatability of the measured performance 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

Appendix F

Price Quotations

digital

COMPAQ
DIGITAL Products and Services

February 2, 2000
Compaq Computer Corporation
Atten: Maria Lopez

Quote No: 131713

Dear Maria:
Thank you for your inquiry. We are pleased to quote as follows.

SERVER HARDWARE:	PART NUMBER	QTY	UNIT PRICE	EXTENSION
Compaq AlphaServer ES40 Model 6/667	DA-64BAA-EA	1	48803	48803
ES40 Tower Enclosure Kit	BA61M-MT	1	350	350
ES40 SMP SPU, 667MHZ, UNIX	KN610-BB	3	8750	26250
5YR 9X5/4HR ES40 TWR/DRW 3-5GB	*FM-6C4HR-60*	*1*	*19165*	*19165*
ES40 2GB Memory Option	MS610-EA	7	19354	135478
5YR 9X5/4HR ES40 1GB MEM ADDER	*FM-6R4HR-60*	*14*	*2706*	*37884*
ES40 Power Supply	H7906-A9	2	875	1750
3-Chan. LVD 64 Bit PCI RAID Controller 16MB mem	DS-KZPCC-CC	9	2660	23940
5YR 5X9/4HR, P/S, ADPT	*FM-PS4HR-60*	*9*	*206*	*1854*
Combo Card; Ethernet/SCSI	FR-PCTBZ-AA	1	504	504
5YRS 5X9/4HR; NTWK $0-499	*FM-N14HR-60*	*1*	*76*	*76*
Ultra 68VHD 10M Cable Assbly	BN37A-10	34	175	5950
D Shf 180W 1 Doc BLW Metro Blu	DS-BA356-JD	34	990	33660
5YR 5X9/4HR, P/S, ADPT	*FM-PS4HR-60*	*34*	*206*	*7004*
4/8GB 4MM SCSI DAT Tape Drive	TLZ09-VA	1	882	882
5YR 5X9/4HR,4MM DAT TAPE DRV	*FM-4M4HR-60*	*1*	*1147*	*1147*
VT510,WHITE,NORTH AMER,NO KEY	VT510-AA	1	277	277
US/CANADA W95 KYBD WHIT	PCXLA-NA	1	18	18
9.1GB 7200RPM UltraSCSI 16bit	DS-RZ1DA-VW	259	455	117845
17-03212-05 8MP-8MP Patch Cable	BN25G-07	6	6	36
SERVER SOFTWARE:				
5YR AS ES40 UNIX BRZ9X5	*FM-62UN9-60*	*1*	*9457*	*9457*
5YR As ES40 Unix SMP	*FM-62USM-60*	*3*	*1121*	*3363*
cVISN MC MG MUL SYS 1 LIC	QM-MQDAA-BA	1	697	697
clearVISN MUL CDRM KIT	QA-5FVAB-H8	1	69	69
Dig UNIX Alpha CDRM	QA-MT4AA-H8	1	277	277
CLIENT HARDWARE:				
Compaq Proliant 1600	153552-001	4	3036	12144

270 Broadway, Hillsdale, NJ 07642 Tel. 201-666-1122 Fax 201-666-1122
www.igsolutions.com

digital

IC Systems Solutions

COMPAQ DIGITAL Products and Services

5YR 5x9 ND Desktop & 17 Or Less Monitor	*FM-3PXHW-60*	*4*	*494*	*1976*
Ethernet Dual Channel	*338456-B21*	*8*	*329*	*2632*
NC3132 Dual 10/100 Module	338456-B22	8	263	2104
256MB SDRAM DIMM Memory Option	313618-B21	8	1132	9056
18GB Drive	388144-B22	4	989	3956
V700 15" Color Monitor	325900-001	4	185	740
CLIENT HARDWARE SUBTOTAL:				
CLIENT SOFTWARE:				
Application Optimizer WNT Server Edition	QB-641AA-SA	4	315	315
Application Optimizer WNT Server Service	*QT-641AA-XA*	5	72	72

The Compaq AlphaServerES40 carries a five (5) year,7X24/4HR response warranty. The storage products have a five (5) year on-site, 4-hour,7-day per week response with a five (5) year return to manufacture warranty. All other products carry a standard warranty of five (5) years on-site:4-hour X 7-days per week.

30% Discount based on total dollar volume: Server Hardware and Software

10% Discount based on total dollar volume: FM part numbers

Valid. This quote is valid for 60days from date

Terms TBD

Delivery 15 Days ARO

Shipping. FOB Origin

Warranty. Manufactures New Equipment

Installation: Included

Sincerely,

Philip K Nolan
IC System Solutions
(201)-666-1122 exten 111
(201)-666-0956 fax
phil@icssolutions.com

Micro Warehouse
YOUR #1 SOURCE FOR COMPUTER PRODUCTS WORLDWIDE.

CORPORATE SALES
1690 OAK STREET, LAKEWOOD, NJ 08701
PHONE: (800) 622-6222 FAX: (732)905-1652

Compaq Computer Corporation
Attn: ***Maria Lopez***
Phone (603) 884-6082
Fax (603) 884-6082

Date: 2/1/2000
Quote# ***852***
Quote valid for 30 days
Sales Quote

QTY	ITEM #	DESCRIPTION	UNIT	TOTAL
3	DEH3552	ETHERFAST 8 PORT 100BASE-TX HUB		
		FEATURES:		
		* CONVENIENT DESKTOP DESIGN		
		* EASY PLUG AND PLAY INSTALLATION		
		* BUILT IN DATA COLLISION CONTR & FRAME RETIMING		
		* UPLINK PORT F/EASY EXPANDABILITY OTHER HUBS	$119.00	$357.00
3481	DEH2924	8 PORT 10BT HUB (8 RJ45 & 1 BNC)		
		FEATURES:		
		* CAN BE CASCADED W/ADDITIONAL HUBS UPLINK PORTS		
		* SUPPORTS CASCADE/DAISY, UP TO 98 PORTS PER LAN SEGM		
		* LED'S FOR LINK STATUS, TRAFFIC AND POWER		
		* AUTO DEVICE MONITORING F/EXCESS NOISE/COLLISIONS		
		* RETIMING AND RESTORATION OF DAMAGE PACKETS	$32.50	$113,132.50

Thank you for your consideration.

Sincerely,

Robert Rumsby (Ext: 20529)
Corporate Account Executive
dc