HP AlphaServer ES45 Client/Server System
using
Sybase Adaptive Server Enterprise® 12.5
and
HP Tru64 UNIX 5.1B

TPC Benchmark®C
Full Disclosure Report

August 15, 2002

First Edition – August 15, 2002

Hewlett-Packard Company believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. Hewlett-Packard Company assumes no responsibility for any errors that may appear in this document.

The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, Hewlett-Packard Company provides no warranty of the pricing information in this document.

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

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

Printed in U.S.A., August 15, 2002

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark© C test conducted on the HP AlphaServer ES45 Model 68/1250 in a client/server configuration, using Sybase Adaptive Server Enterprise 12.5. The operating system was HP Tru64 UNIX V5.1B.

TPC Benchmark© C Metrics

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

Standard and Executive Summary Statements

Page 4 begins the executive summary of the benchmark results for the HP AlphaServer ES45 Model 68/1250 system. The Standard System Summary is given below.

Company Name	System Name	Database Software	Operating System Software
Hewlett-Packard Company	HP AlphaServer ES45 4 CPU Client/Server System	Sybase Adaptive Server Enterprise 12.5	HP Tru64 UNIX V5.1B
Total System Cost	**TPC-C Throughput**	**Price Performance**	**Availability Date**
$531,965	56,375	$9.44	September 27, 2002

Auditor

The benchmark configuration, environment and methodology used to produce and validate the test results, and the pricing model used to calculate the cost per tpmC©, were audited by Francois Raab of Infosizing to verify compliance with the relevant TPC specifications.

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

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

or

Hewlett-Packard Company
Systems Quality and Performance Engineering
110 Spit Brook Road, ZKO2-3/M31
Nashua, NH 03062 USA

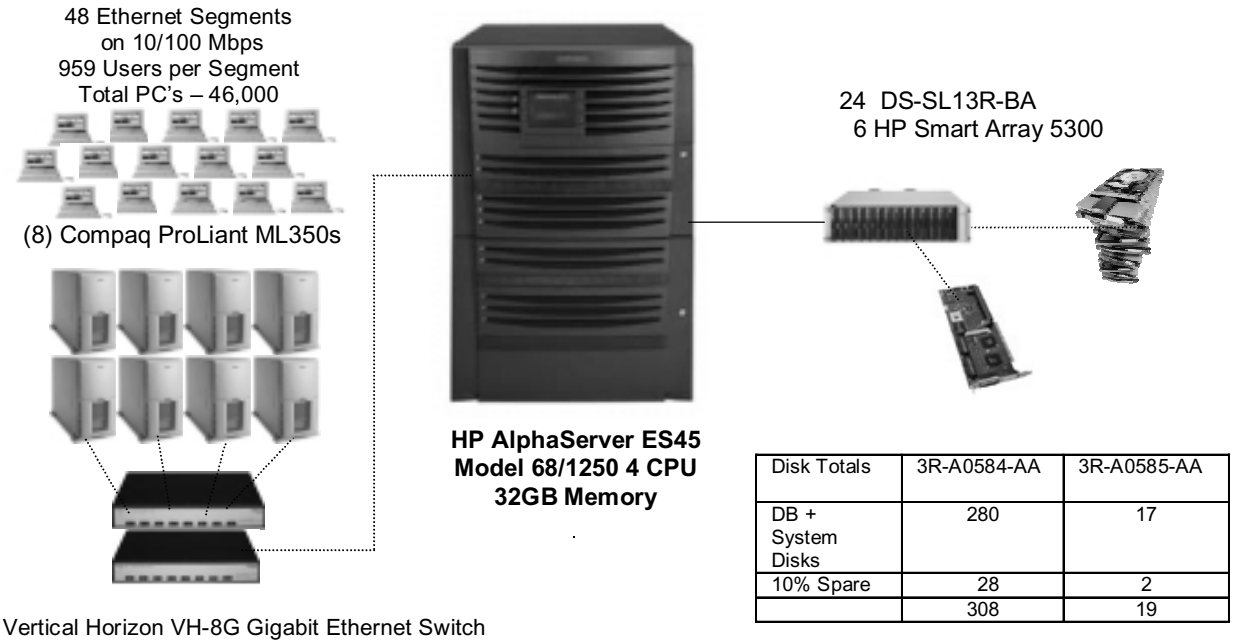
HP AlphaServer ES45 Model 68/1250 4 CPU Client/Server System

TPC-C Rev. 5

Report Date: August 15, 2002

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$531,965	56,375	$9.44	September 27, 2002

Processors	Database	Operating System	Other Software	Number of Users
(4) 1250 MHz Alphachip 21264C	Sybase Adaptive Server Enterprise 12.5	HP Tru64 UNIX V5.1B	BEA Tuxedo 6.5 CTS	46,000



Disk Totals	3R-A0584-AA	3R-A0585-AA
DB + System Disks	280	17
10% Spare	28	2
	308	19

	System Components Database Server, HP AlphaServer ES45		Client Components (Each of 8) Compaq ProLiant ML350	
	Qty	Type	Qty	Type
Processors	4	68/1250 MHz Alphachip 21264C	2	1GHz Intel Pentium III
Cache Memory	16	MB Cache	1	512KB Cache
Memory	4	8 GB Memory	1	128MB Memory
Total Memory	32	GB Memory	2	256MB Memory
Disk Controllers	6	HP Smart Array 5300		640MB Total Memory
Disks	280	9.1 GB Disks	1	18GB Disk
	17	18 GB Disks		
Total Disk Storage	2.85	TB		

Description	Part Number	Third Party Brand	Unit Price Pricing	Qty	Extended Price	3 yr. Maint. Price
Server Hardware						
ES45 68/1250 M2 4GB UNIX	DA-68EBA-DA	1	53,456	1	53,456	17,896
ES45 Tower Enclosure	BA61M-CT	1	325	1	325	Inc.
ES45 68/1.25GHZ SMP CPU Unix	KN610-EB	1	11,050	3	33,150	Inc.
ES45 4GB Memory Option	MS620-DA	1	17,306	7	121,142	Inc.
PCI to GbE SX Adapter	DEGPA-SA	1	1,287	1	1,267	Inc.
Power Supply, Self Sensing	H7906-A9	1	813	2	1,626	Inc.
SmartArray 5304A/128 for Alpha	3X-KZPDC-DF	1	1,624	6	9,744	Inc.
Ultra 68VHD 10M Cable Assembly	BN37A-03	1	91	24	2,184	0
Star Lite Storage Shelf	DS-SL13R-BA	1	2,290	24	54,960	0
12/24gb 4MM Dat 5.25 Tape Drive	TLZ10-LB	1	444	1	444	862
9GB 10K U3 UNI HP HARD DRIVE **	3R-A0584-AA	1	207	308	63,756	Spared
18GB 10K U3 UNI HP HARD DRIVE **	3R-A0585-AA	1	207	19	3,933	Spared
VT510,WHITE,NORTH AMER,NO KEY	VT510-AA	1	397	1	397	0
US& Canada/English KB	PCXLA-NA	1	18	1	18	0
17-03212-05 8mp-8mp Patch Cable	BN25G-07	1	6	1	6	0
Vertical Horizon VH-8G Gigabit Ethernet Switch	VH-8G	2	2,747.25	4	10,989	
				Subtotal	**357,397**	**18,758**
Server Software						
3YR, As ES40/45 Unix Brnz24x7	*FM-E4WUS-36*	*1*	*1,896*	*1*		*1,896*
3YR AS ES45 Unix SMP	*FM-62USM-36*	*1*	*214*	*3*		*642*
3YR Digital UNIX O/S & LP	*FM-CDDST-36*	*1*	*6,765*	*1*		*6,765*
Tru64 UNIX Alpha CDRM	QA-MT4AA-H8	1	293	1	293	
Sybase ASE	Sybase	3	34,995	1	34,995	22,100
				Subtotal	**35,288**	**31,403**
Client Hardware						
Compaq ProLiant ML350 1.0Ghz	225864-001	1	1,587	8	12,696	9,280
Pentium II P1000-256k Processor	207068-B21	1	571	8	4,568	Inc.
512MB SDRAMM DIMM Memory	128279-B21	1	435	8	3,480	Inc.
ProLiant ML350 Hot Plug Drive Cage	161275-B21	1	413	8	3,304	Inc.
18.2GB Wide Ultra 3 Drive	3R-A0585-AA	1	207	8	1,656	Inc.
NC3134 Fast Ethernet NIC 64 PCI Dual Port 10/100	138603-B21	1	207	16	3,312	Inc.
NC3135 Fast Ethernet Mod Dual 10/100 Upgrade for NC3134	138604-B21	1	207	8	1,656	Inc.
Gigabit Daughter CD Upgrade	338456-B23	1	516	8	4,128	Inc.
128MB SDRAM Dimm Memory Option	313615-B21	1	144	8	1,152	Inc.
SC-SC Dual FO Cbl, MM, PP, 4.5M	BN34B-4E	1	62	8	496	Inc.
V700 15" Monitor NH US	261602-001	1	130	10	1,300	Spared
				Subtotal	**37,748**	**9,280**
Client Software						
Sybase Software Developers Kit	Sybase	3	1,645	1	1,645	1,086
BEA Tuxedo 6.5 Tier 1	BEA Tuxedo	4	2,850	8	22,800	16,560
				Subtotal	**24,445**	**17,646**

Notes: ** 10% Spares**

Subtotal	**$454,878**	**$77,087**
Three Year Cost of Ownership		**$531,965**
1=IC System Solutions, 2=Enterasys Networks, 3=Sybase, 4=BEA Tuxedo	**tpmC Rating:**	**56,375**
Audited by InfoSizing	**$ / tpmC:**	**$9.44**

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

Numeric Quantities Summary
AlphaServer ES45 Client/Server

MQTH

MQTH, computed Maximum Qualified Throughput	56,375

Response Times in seconds

Transaction	90th percentile	Average	Maximum
New-Order	1.334	0.758	6.127
Payment	1.395	0.864	300.138
Order-Status	1.337	0.765	5.882
Delivery (interactive)	1.180	0.503	15.542
Delivery (deferred)	26.937	7.471	41.265
Stock-Level	1.572	0.916	6.573
Menu	0.396	0.225	4.670

Transaction Mix in percent of total transactions

Transaction	Total Occurrences	Percentage
New-Order	6765043	44.975%
Payment	6468907	43.006%
Order-Status	603301	4.011%
Delivery (interactive)	602342	4.004%
Stock-Level	602221	4.004%

Keying/Think Times in seconds

Transaction	Minimum	Average	Maximum
New-Order	18.000/0.000	18.004/12.005	18.089/119.990
Payment	3.000/0.000	3.000/12.002	3.046/119.996
Order-Status	2.000/0.000	2.000/10.004	2.034/ 99.990
Delivery (interactive)	2.000/0.000	2.000/ 5.005	2.027/ 49.817
Stock-Level	2.000/0.000	2.000/ 5.007	2.034/ 49.757

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

Ramup up time	30:00:00 minutes
Measurement interval	120:00.00 minutes
Number of checkpoints	4
Number of New Order transactions	6765043
Number of transactions (all types) completed in measurement interval	15041814

Table of Contents

TPC Benchmark C Full Disclosure

The *TPC Benchmark C Standard Specification* requires test sponsors to publish, and make available to the public, a full disclosure report for the results to be considered compliant with the Standard. The required contents of the full disclosure report are specified in Clause 8. This report is intended to document 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,*
- *checkpoint interval in minutes,*
- *number of transactions (all types)completed within the measurement interval, computed maximum Qualified Throughput in tpmC,*
- *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 HP C application code and the Sybase anonymous block SQL code.

1.5 Sponsor

A statement identifying the benchmark sponsor(s) and other participating companies must be provided.
This benchmark test was sponsored by Hewlett-Packard Company and Sybase Inc., and attested to by InfoSizing.

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/commit options.*
- *Recovery/locking options*
- *Operating system and application configuration parameters.*
- *Compilation and linkage options and run-time optimizations used to create/install applications, OS, and/or databases.*

The Test Sponsor has elected to provide a listing of all parameters and options. Appendix D contains the tunable parameters used in the TPC-C tests.

1.7 Configuration

Provide diagrams of both the measured and priced configuration, 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.*
- *Type and the run-time execution location of software components (e.g., DBMS, client processes, transaction monitors, software drivers, etc.)*

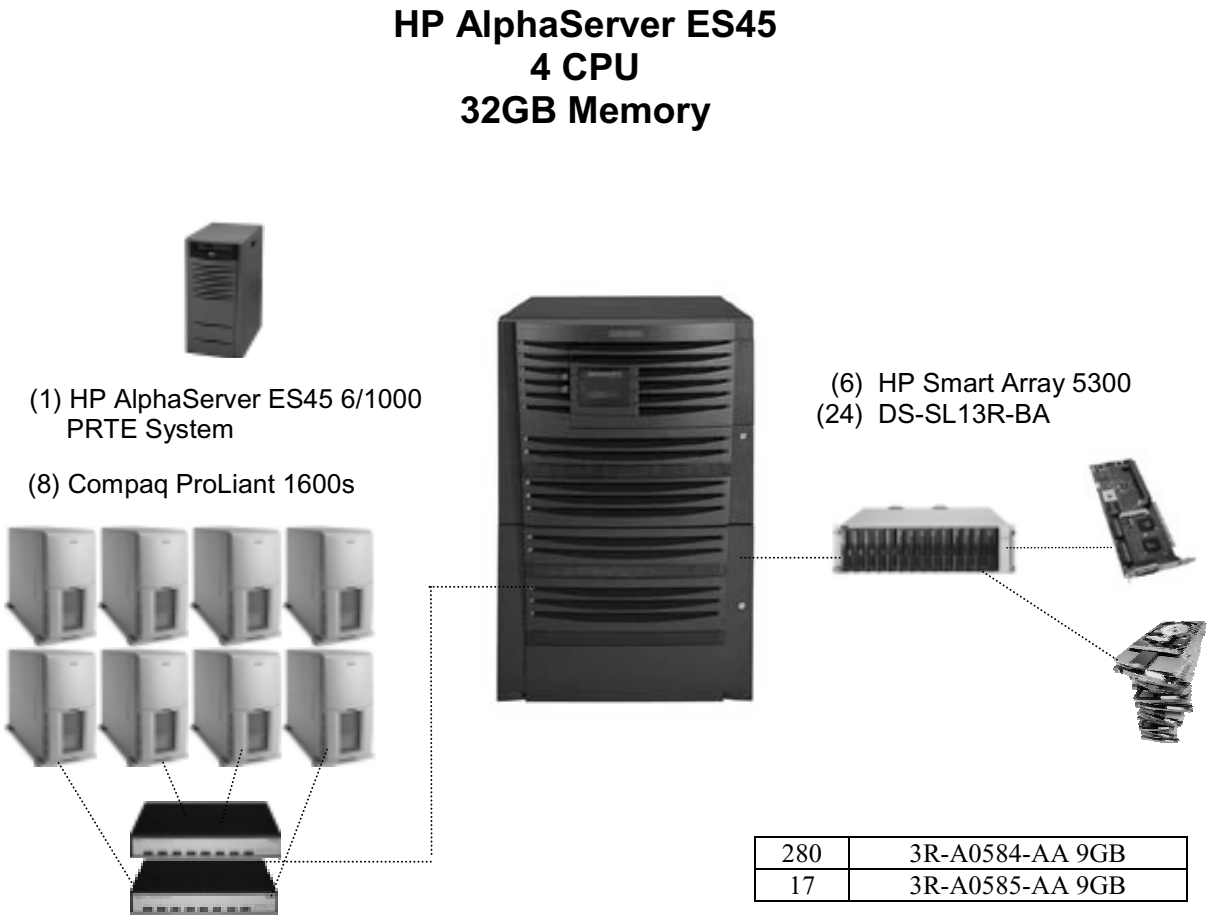
The TPC-C terminal users were emulated by HP's Portable Remote Terminal Emulator (PRTE) software, which ran on one HP AlphaServer ES45 6/1000 computer. Each emulated terminal was connected using HTTP (HyperText Transport Protocol) to a client node.

The clients consisted of 8 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 Enterprise 12.5 using dblib calls.

The system consisted of 1 AlphaServer ES45 with 4 CPUs and 32GB of memory. The measured server was configured (17) 3R-A0585-AA 18GB, and (280) 3R-A0584-AA 9GB. The priced configuration consists of (19) A0585-AA 18GB, and (308) 3R-A0584-AA 9GBb which includes an additional 10% spareable disks.

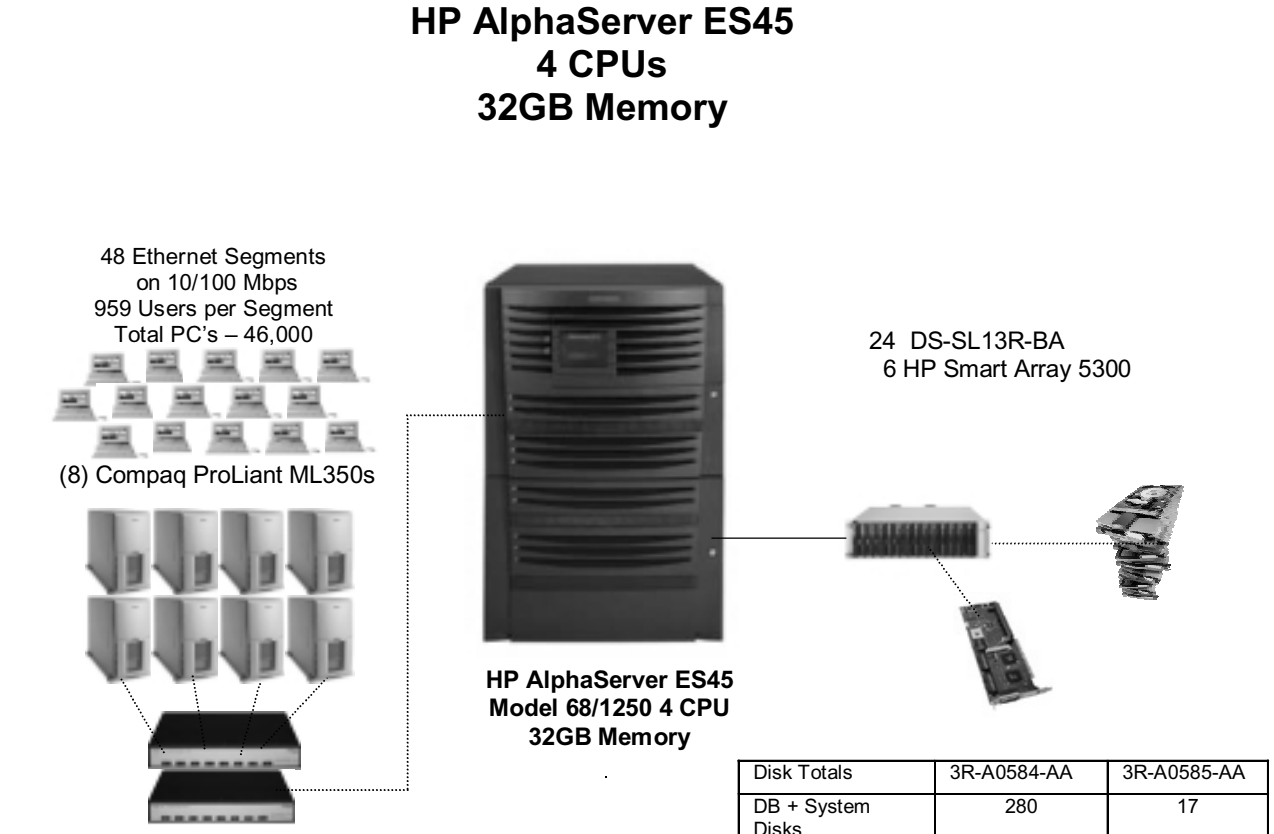
Measured Configuration

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 Configuration

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



Disk Totals	3R-A0584-AA	3R-A0585-AA
DB + System Disks	280	17
10% Spare Disks	28	2
Total	308	19

2. Logical Database Design Related Items

2.1 Table Definitions

Listings must be provided for all table definition 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.

The scripts used to build and load the database are 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.

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 were 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 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 unused item number must be disclosed.

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

3.7 Number of Order-Lines

The number of items per orders entered by 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 were 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)	84.97%
% remote Payment transactions	3.8	(14.0 - 16.0)	15.03%
% non-primary key access, Payment	3.9	(57.0-63.0)	60.02%
% non-primary key access, Order Status	3.9	(57.0-63.0)	59.89%
% 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 in percent of total transactions

Transaction	Total Occurrences	Percentage
New-Order	6765043	44.975%
Payment	6468907	43.006%
Order-Status	603301	4.011%
Delivery (interactive)	602342	4.004%
Stock-Level	602221	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 to 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 4,600 warehouse database was used for the Consistency and system crash.

Case A was followed for the execution of Isolation Test 7, as described in the TPC-C Standard Specification, Clause 3.4.2.7.

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 3.3.3 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).*

Tests were conducted for each of the preceding types of failures and successfully demonstrated that the durability properties were met.

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 4,600 warehouses database using 46,000 emulated terminals.

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 at least 5 minutes. Then the system was powered off.
3. The test was aborted on the RTE.
4. The system was powered back on and rebooted.
5. Sybase Adaptive Server Enterprise 12.5 was restarted and recovered the database from the transaction log.

6. 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 RTE.
7. Several orders recorded by the RTE were checked in the database to make sure they existed.
8. 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	4,600
District	46,000
Customer	138,000,000
History	138,000,000
New-Order	41,400,000
Order	138,000,000
Order-Line	1,380,000,000
Stock	460,000,000
Item	100,000

5.2 Distribution of Tables and Log

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

The following benchmark configuration tables over all the disks of the priced system is an extension of the distribution in the tested system. System was configured with enough mirrored log disk capacity to support 8 hours of run time.

Distribution of Data on the AlphaServer ES45

cidx02	/dev/rdisk/dsk111d	33.33%	ordlne06	/dev/rdisk/dsk124d	12.50%
cidx03	/dev/rdisk/dsk122d	33.33%	ordlne07	/dev/rdisk/dsk131d	12.50%
cust01	/dev/rdisk/dsk105b	6.66%	ordlne08	/dev/rdisk/dsk170d	12.50%
cust02	/dev/rdisk/dsk111b	6.66%	stock01	/dev/rdisk/dsk101b	4.00%
cust03	/dev/rdisk/dsk122b	6.66%	stock02	/dev/rdisk/dsk108b	4.00%
cust04	/dev/rdisk/dsk129b	6.66%	stock03	/dev/rdisk/dsk169b	4.00%
cust05	/dev/rdisk/dsk136b	6.66%	stock04	/dev/rdisk/dsk126b	4.00%
cust06	/dev/rdisk/dsk136d	6.66%	stock05	/dev/rdisk/dsk133b	4.00%
cust07	/dev/rdisk/dsk129d	6.66%	stock06	/dev/rdisk/dsk169d	4.00%
cust08	/dev/rdisk/dsk106d	6.66%	stock07	/dev/rdisk/dsk126d	4.00%
cust09	/dev/rdisk/dsk106b	6.66%	stock08	/dev/rdisk/dsk133d	4.00%
cust10	/dev/rdisk/dsk112b	6.66%	stock09	/dev/rdisk/dsk102b	4.00%
cust11	/dev/rdisk/dsk123b	6.66%	stock10	/dev/rdisk/dsk109b	4.00%

cust12	/dev/rdisk/dsk130b	6.66%	stock11	/dev/rdisk/dsk120b	4.00%
cust13	/dev/rdisk/dsk123d	6.66%	stock12	/dev/rdisk/dsk127b	4.00%
cust14	/dev/rdisk/dsk137b	6.66%	stock13	/dev/rdisk/dsk134b	4.00%
cust15	/dev/rdisk/dsk130d	6.66%	stock14	/dev/rdisk/dsk120d	4.00%
history	/dev/rdisk/dsk137d	100.00%	stock15	/dev/rdisk/dsk127d	4.00%
log01	/dev/rdisk/dsk173b	50.00%	stock16	/dev/rdisk/dsk134d	4.00%
log02	/dev/rdisk/dsk173d	50.00%	stock17	/dev/rdisk/dsk103b	4.00%
master	/dev/rdisk/dsk114d	100.00%	stock18	/dev/rdisk/dsk110b	4.00%
orders01	/dev/rdisk/dsk114b	33.33%	stock19	/dev/rdisk/dsk121b	4.00%
orders02	/dev/rdisk/dsk125b	33.33%	stock20	/dev/rdisk/dsk128b	4.00%
orders03	/dev/rdisk/dsk132b	33.33%	stock21	/dev/rdisk/dsk135b	4.00%
ordlne01	/dev/rdisk/dsk107b	12.50%	stock22	/dev/rdisk/dsk135d	4.00%
ordlne02	/dev/rdisk/dsk113b	12.50%	stock23	/dev/rdisk/dsk121d	4.00%
ordlne03	/dev/rdisk/dsk124b	12.50%	stock24	/dev/rdisk/dsk128d	4.00%
ordlne04	/dev/rdisk/dsk131b	12.50%	stock25	/dev/rdisk/dsk104b	4.00%

5.3 Type of Database

A statement must be provided that describes:
The data model implemented by the DBMS used (e.g., relational, network, hierarchical)

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 12.5 from Sybase, Inc. Sybase Adaptive Server Enterprise 12.5 is a relational DBMS. SQL stored procedures were used and were invoked from the clients using dblib calls. The database was built for 4,600 warehouses.

5.4 Database Partitions/Replications

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

No tables were used for partitioning in this benchmark test.

5.5 60-Days Space Requirement

Details of the 60-day space calculations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-line, and History) must be disclosed.

Warehouses	4600	**tpmC**	56375	**tpmC/W**	12.26	
Table	**Rows**	**Data**	**Index**	**5% Space**	**8H Space**	**Total Space**
Warehouse	4,600	452	4	23		479
District	46,000	4,720	16	237		4,973
Item	100,000	9,304	28	187		9,519
New-order	41,400,000	662,404	1,980		92,000	756,384
History	138,000,000	7,728,160	0		1,265,815	8,993,975
Orders	138,000,000	3,950,012	11,764		648,910	4,610,686
Customer	138,000,000	92,184,004	6,625,508	1,976,190		100,785,702

Order-line	1,380,000,000	90,649,496	274,068		14,892,608	105,816,172
Stock	460,000,000	141,680,004	386,056	2,841,321		144,907,381
Totals		336,868,556	7,299,424	4,817,958	16,899,333	365,885,271

Segment	LogDev Cnt.	Seg. Size	Needed	Overhead	Not Needed
wdino	1	1,203,200	779,068	7,791	416,342
history	1	11,776,000	9,083,915	90,839	2,601,246
order	3	5,885,952	4,656,792	46,568	1,182,592
customer	18	103,388,160	101,793,559	1,017,936	576,665
order_line	8	117,760,000	106,874,333	1,068,743	9,816,923
stock	25	150,732,800	146,356,455	1,463,565	2,912,780
Totals		390,746,112	369,544,123	3,695,441	17,506,548

Dynamic space	99,564,821	Sum of Data for Order, Order-Line and History (excluding free extents)
Static space	253,116,558	Data + Index + 5% Space + Overhead - Dynamic space
Free space	20,558,186	Total Seg. Size - Dynamic Space - Static Space - Not Needed
Daily growth	19,523,363	(Dynamic space/W * 62.5)* tpmC
Daily spread	(8,726,859)	Free space - 1.5 * Daily growth (zero if negative)
60 day (KB)	1,424,518,321	Static space + 60 (daily growth + daily spread)
60 day (GB)	1358.53	Excludes OS, Paging and RDBMS Logs
Log per N-O txn	2	Number of 2K blocks per New-Order transaction
8 Hour Log (GB)	103.23	

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

60 day (GB)	1,358.53
OS + Sybase + Swap	17.36
	1,375.89

Space Allocated for Benchmarked Configuration (Excluding logs)

Drive Type	Number	Cap. (GB)	Total Capacity (GB)	Needed Space	Space Need
3R-A0584-AA	280	8.67	2167.50		
			2167.50	(792)	0

Space Allocated for Logs

Drive Type	Number	Cap. (GB)	Capacity (GB)	
3R-A0585-AA	16	17.36	277.76	unmirrored capacity
3R-A0585-AA	8	17.36	138.88	mirrored capacity

Space Needed for Logs	103.225708	
Total Priced devices		
3R-A0585-AA	17	
3R-A0584-AA	280	
Spares	2	
Spares	28	
Total Priced devices	327	mirrored logs + space allocated 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 5.2 through 5.10. The measured tpmC for the HP AlphaServer ES45 C/S configuration was 56,375 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	1.334	0.758	6.127
Payment	1.395	0.864	300.138
Order-Status	1.337	0.765	5.882
Delivery (interactive)	1.180	0.503	15.542
Delivery (deferred)	26.937	7.471	41.265
Stock-Level	1.572	0.916	6.573
Menu	0.396	0.225	4.670

6.3 Think and Keying Times

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

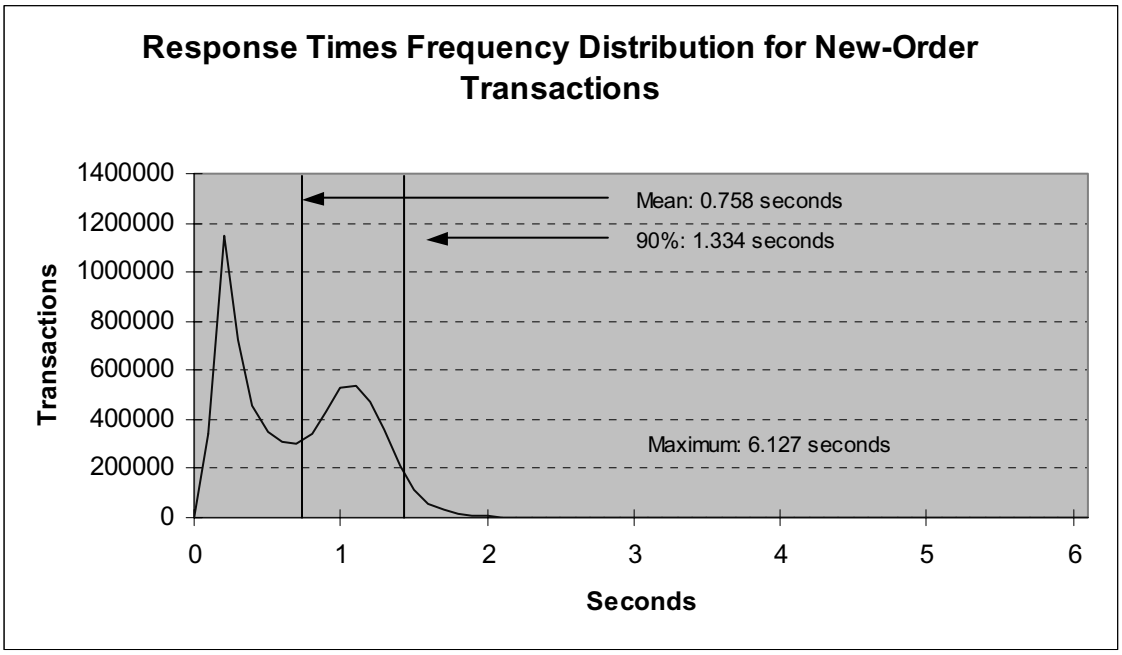
Keying/Think Times in seconds

Transaction	Minimum	Average	Maximum
New-Order	18.000/0.000	18.004/12.005	18.089/119.990
Payment	3.000/0.000	3.000/12.002	3.046/119.996
Order-Status	2.000/0.000	2.000/10.004	2.034/ 99.990
Delivery (interactive)	2.000/0.000	2.000/ 5.005	2.027/ 49.817
Stock-Level	2.000/0.000	2.000/ 5.007	2.034/ 49.757

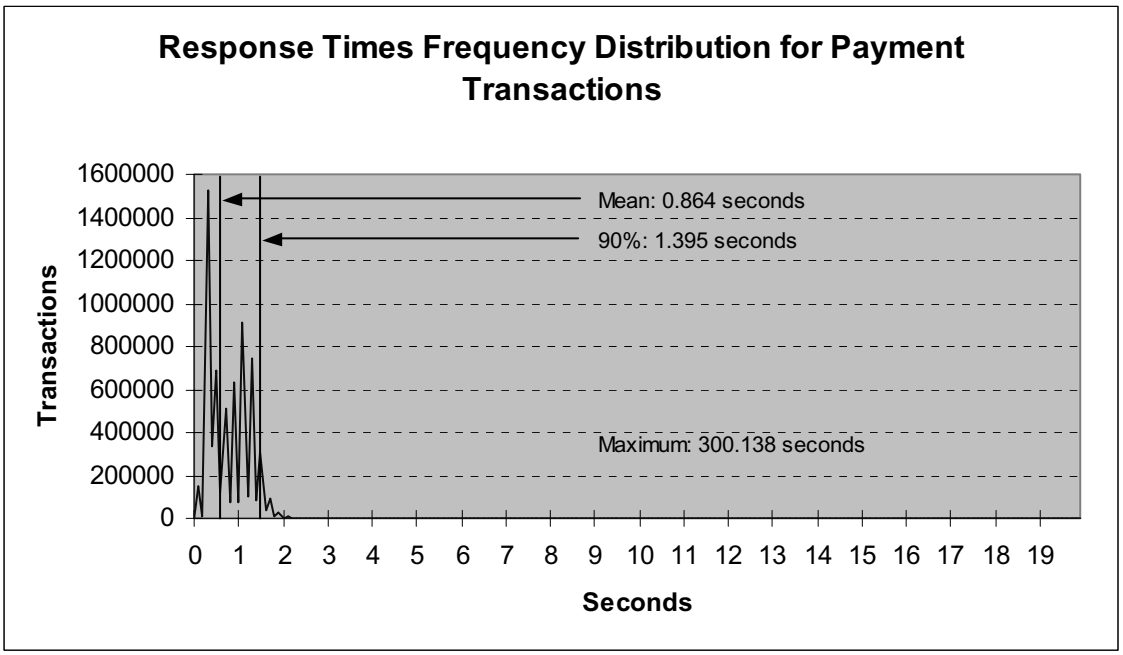
6.4 Response Times Frequency Distribution

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

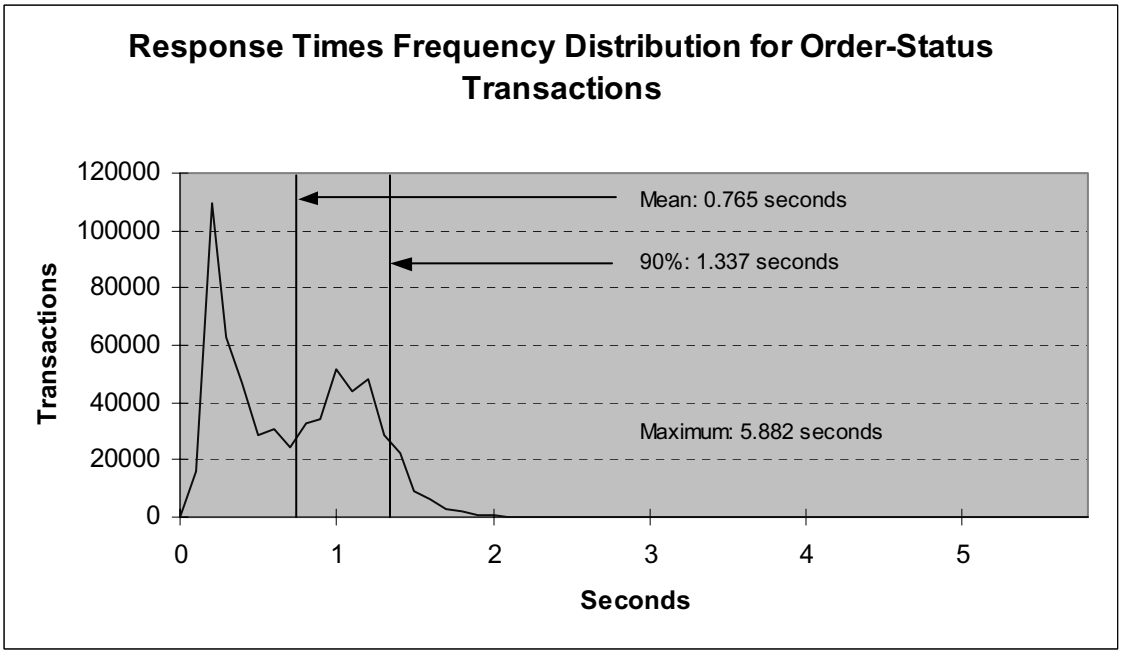
Response Times Frequency Distribution for New-Order Transactions



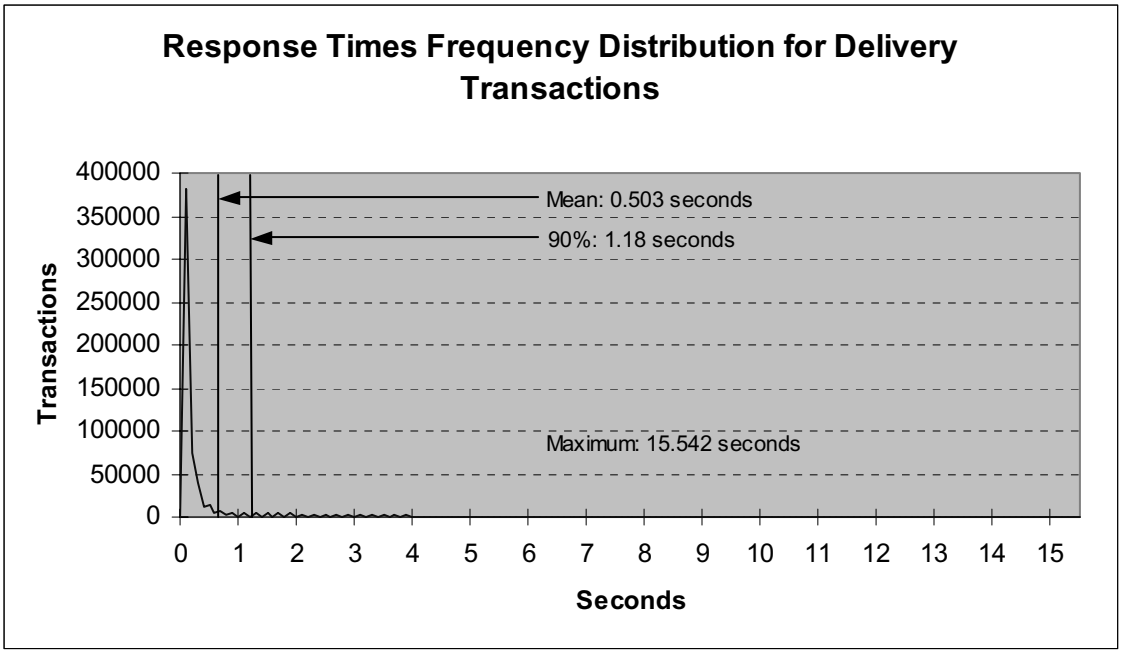
Response Times Frequency Distribution for Payment Transactions



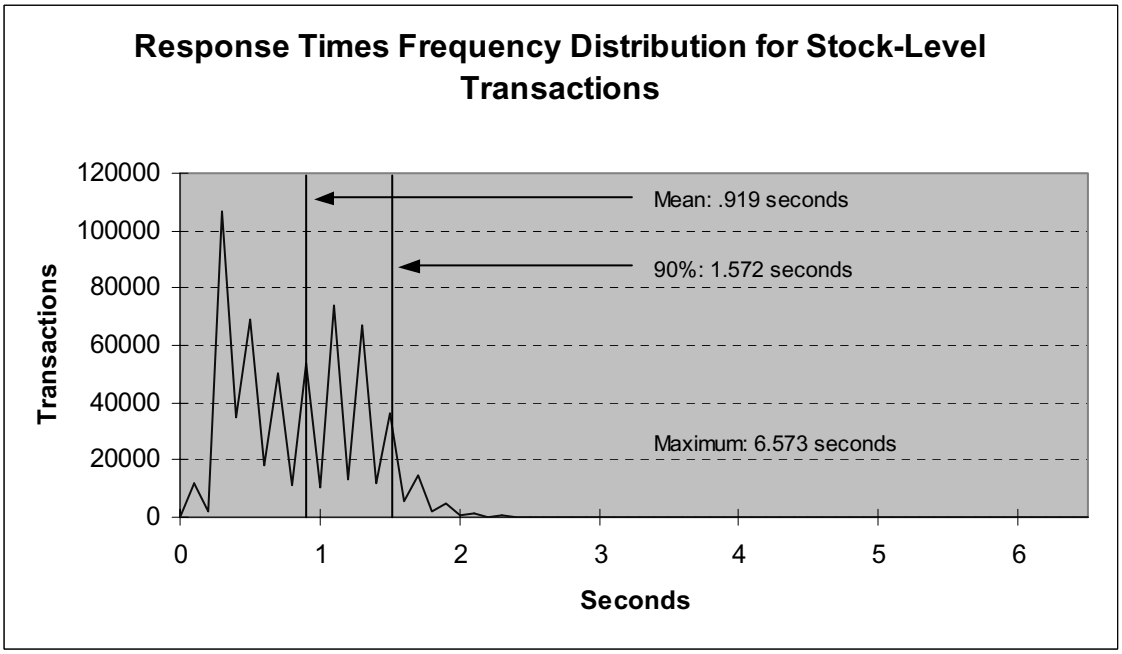
Response Times Frequency Distribution for Order-Status Transactions



Response Times Frequency Distribution for Delivery Transactions



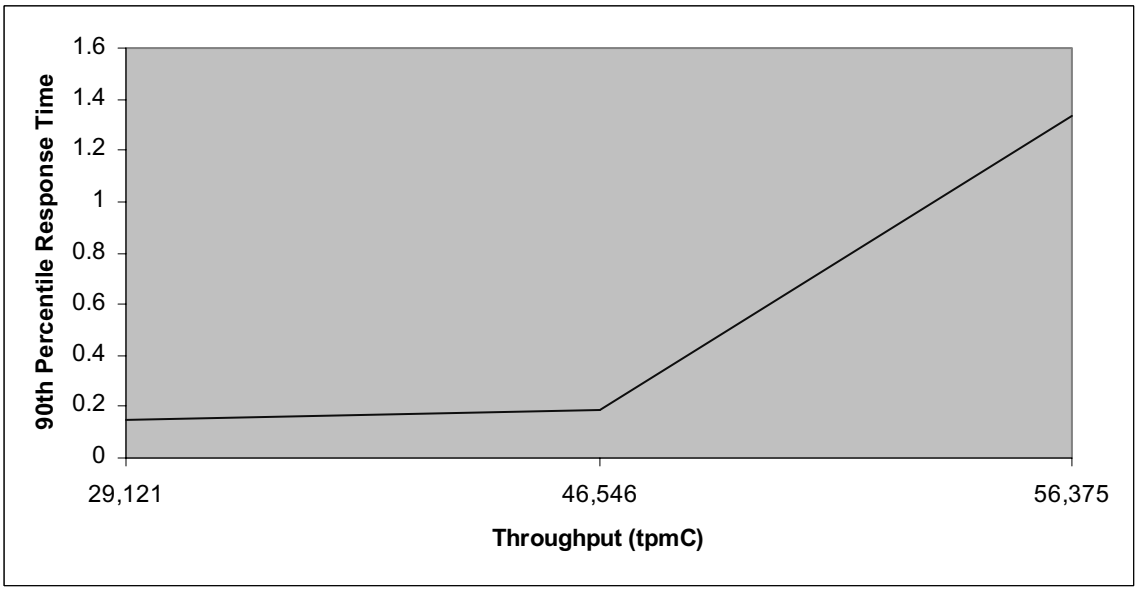
Response Times Frequency Distribution for Stock-Level Transactions



6.5 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.

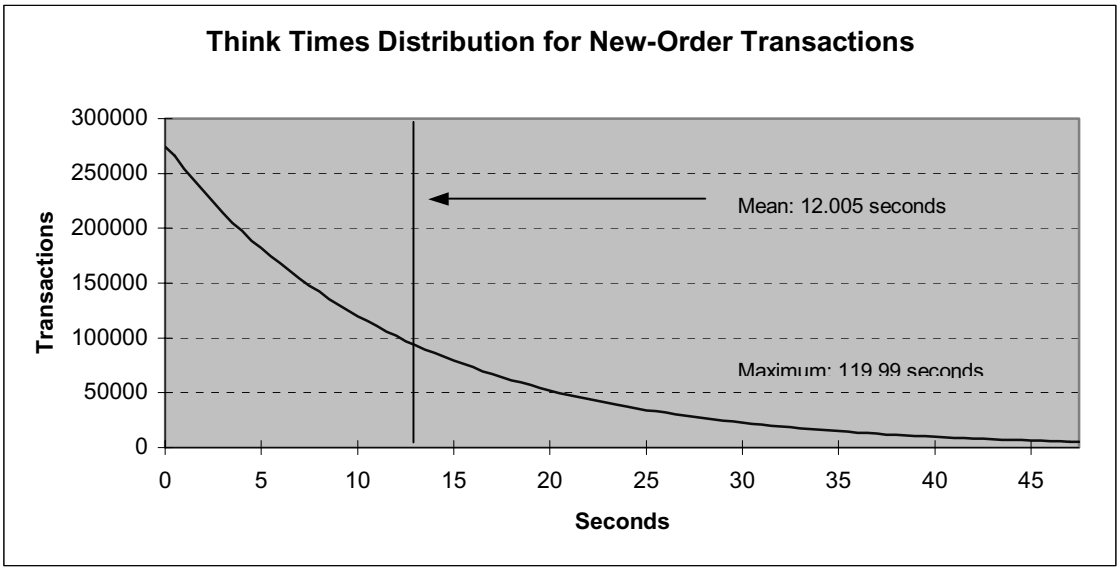
New-Order Response Time versus Throughput



6.6 Think Times Frequency Distribution

Think times frequency distribution curves (see Clause 5.6.3) must be reported for the New Order Transaction.

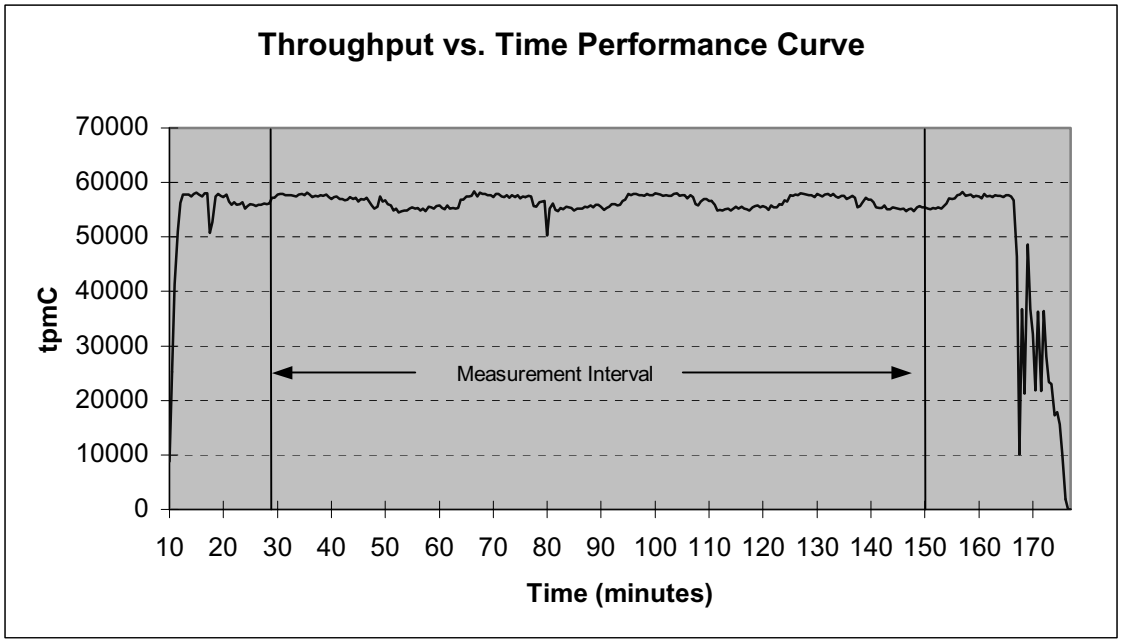
Think Times Distribution for New-Order Transactions



6.7 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.

Throughput vs. Time Performance Curve



6.8 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 6.7 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.

6.8.1 Transaction Flow

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

BEA Tuxedo 6.5 CTS for Windows 2000 server 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 1000 MB Hub using Sybase Adaptive Server Enterprise 12.5 dblib calls. BEA Tuxedo 6.5 CTS routes the transaction and balances the load according to values in the registry governing the number of processing procedure servers. Following is the

transaction flow description.

- 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. Each of the back-end processing procedures can process 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 BEA Tuxedo 6.5 CTS 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 BEA Tuxedo 6.5 CTS transaction manager.
- BEA Tuxedo 6.5 CTS passes the workspace back to the TPC-C main front-end.

6.8.2 Database Transaction

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

Using Sybase Adaptive Server Enterprise 12.5 dblib calls, the TPC-C back-end program interacts with Sybase Adaptive Server Enterprise 12.5 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 Enterprise 12.5 proceeds to update the database as follows:

When Sybase Adaptive Server Enterprise 12.5 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 Enterprise 12.5 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. For each of the TPC Benchmark C transaction types, the following steps are executed.

BEA Tuxedo 6.5 CTS for Windows 2000 server 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 BEA Tuxedo 6.5 CTS 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. Each of the back-end processing procedures can process 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 BEA Tuxedo 6.5 CTS 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 BEA Tuxedo 6.5 CTS transaction manager.
- BEA Tuxedo CTS 6.5 passes the workspace back to the TPC-C main front-end.

6.9 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 application on the desired client. The application 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 application on the client is run under the control of the Internet Information Server.

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 Sybase's in-memory data cache for a while before being written to disk.

6.10 Determining Reproducibility

Duration Of the Checkpoint Interval

The start time and duration in seconds of at least four (4) longest checkpoints during the Measurement Interval must be disclosed.

Appendix D contains the report for the checkpoints.

6.11 Duration of Measurement Period

Each experiment was run for a minimum of 150 minutes. The Measurement Interval was 120 minutes and started approximately 30 minutes after all simulated users had begun executing transactions with a 10 minute ramp down.

6.12 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. The RTE adjusted the transaction mix dynamically in accordance with clause 5.4.2.1.

6.13 Percentage of the Total Mix

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

See Section 3.10.

6.14 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.

6.15 Average Number of Order-Lines

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

See Section 3.10.

6.16 Percentage of Remote Order-Lines

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

See Section 3.10.

6.17 Percentage of Remote Payment Transactions

The percentage of remote Payment transactions must be disclosed.

See Section 3.10.

6.18 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.

6.19 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.

6.20 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 HP AlphaServer ES45 C/S system was set up to checkpoint within every 30 minutes. One checkpoint occurs during the warm-up period, and 4 checkpoints occurred 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.

7.2 Lost Terminal Connections

The number of terminal connections lost during the Measurement Interval must be disclosed

There were no lost connections during the warm-up or Measurement Interval.

7.3 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, 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 front-end clients using HTTP (HyperText Transfer Protocol) over TCP/IP.

7.4 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.

7.5 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 measured configuration, both the client machines (Compaq ProLiant 1600 and HP AlphaServer ES45) are connected into two Vertical Horizon VH-8G Gigabit Ethernet Switch. The emulated PC's are connected on 10/100 Ethernet LANs. The total number of PCs are 46,000.

7.6 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 network between the front-end clients and the server has a bandwidth of 1000MB/s. The network connecting the PCs into 100MB Hub is 100 Mbps.

7.7 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.

The total 3-year price of the entire configuration must be reported, include: hardware, software and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed.

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

8.1.1 Hardware Pricing

HP 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 3 year warranty and 7 X 24 with 4 hour response requirements.

The hardware prices for the HP AlphaServer ES45 4 CPUs 32GB memory and their associated components (e.g., memory, storage subsystem, etc.) are based on IC System Solutions and Enterasys Networks price quotations.

8.1.2 Software Pricing

The priced system uses the following software products:

- HP Tru64 UNIX operating system
- BEA Tuxedo 6.5 CTS

- HP C compiler
- Sybase Adaptive Server Enterprise 12.5
- Microsoft 2000 Server Operating System

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

The Sybase software pricing contact is: Prasanta Gosh, Performance Manager, Sybase, Inc., 5000 Hacienda Drive, Dublin, CA 94568.

8.1.3 Warranty Pricing

In addition to the base warranty, additional warranty has been priced to satisfy the 3-year TPC-C warranty requirements of all products.

The license purchase includes 1 year of warranty service. Two years of additional software warranty is provided for a total of 3 years extended warranty. The software warranty and service levels are the same as the service level for the hardware system on which the software operates.

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 HP AlphaServer ES45 is currently available. The HP Smart Array 5300s are available September 27, 2002. HP Tru64 UNIX 5.1B is available September 27, 2002. Sybase Adaptive Server Enterprise 12.5 is currently available.

8.3 Performance and Price/Performance

A statement of the measured tpmC, as well as the respective calculations for 3-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
HP AlphaServer ES45 Model 68/1250	4 CPU	HP Tru64 UNIX 5.1B Sybase Adaptive Server Enterprise 12.5	56,375	$9.44/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 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 InfoSizing for the test described in this report.

Appendix A

admin.c

```
/*+*************************************************************
**************
 *                                                              *
 *  COPYRIGHT (c) 1997, 2000 BY                                 *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.              *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                                *
 *                                                              *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                              *
 *                                                              *

**************************************************************
*************-*/
/*+          FILE:                       ADMIN.C
 *                                                  Copyright Digital, 1996-
1997
 *
 *          PURPOSE:  Perform administrative tasks such as update the
 *                                      registry, read performace counters,
control services,
 *                                      etc.
 *
 *          Author:                     Bill Carr
 *
            carr@perfom.enet.dec.com
 */

#define WIN32_LEAN_AND_MEAN 1
#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
#include <assert.h>
#include <pdh.h>
#include <httpext.h>

#include <aui_web_srv.h>

#define FILENAMESIZE 1024

PFN_GETEXTENSIONVERSION pVersionFunc = NULL;
PFN_HTTPEXTENSIONPROC   pHttpFunc = NULL;
HINSTANCE                       hInstance = NULL;
static char                 szModName[FILENAMESIZE] = { '\0' };
#define ERROR_NOT_RECOGNIZED "ERROR - Command not
recognized: %s"
#define ERRSTRSIZ 256

BOOL FAR PASCAL AssertShutdown(BOOL fEnable);

BOOL        GetKeyValue( char *query, char *key, char *value );
DWORD       GetPerformanceCounter( EXTENSION_CONTROL_BLOCK
*pECB );
DWORD       GetPerformanceCounterValue( char *machine, char *object,
char *instance,
                                                    char *parent, DWORD
index, char *counter,
                                                    int *count );
DWORD       MainPage( EXTENSION_CONTROL_BLOCK *pECB );
DWORD       ServiceControl( EXTENSION_CONTROL_BLOCK *pECB );
DWORD       SetRegistry( EXTENSION_CONTROL_BLOCK *pECB );
LONG        GetRegistryValue( char *key, char *value, DWORD *ptype,
                                    LPBYTE data, DWORD *psize );
LONG        SetRegistryValue( char *key, char *value, DWORD type,
                                    LPBYTE data, DWORD size );
DWORD       ShutdownSystem( EXTENSION_CONTROL_BLOCK *pECB
);
void        SysErrResponse( EXTENSION_CONTROL_BLOCK *pECB,
char *szErrStr );
_



 char                                   AdminPageFmt[] = \
            "<h1>%s</h1> \
            Status: %d<br> \
            Message: %s<br> \
            <hr> \
            %s \
            <hr> \
            <form method=GET action=%s> \
            <input type=submit name=CMD
value=GetPerformanceCounter> \
            <input type=submit name=CMD value=ServiceControl> \
            <input type=submit name=CMD value=GetRegistry> \
            <input type=submit name=CMD value=SetRegistry> \
            <input type=submit name=CMD value=ShutdownSystem> \
            </form> \
            ";


BOOL FAR PASCAL AssertShutdown(BOOL fEnable)
{
  HANDLE hToken;
  LUID ShutdownValue;
  TOKEN_PRIVILEGES tkp;



  if (!OpenProcessToken(GetCurrentProcess(),
       TOKEN_ADJUST_PRIVILEGES | TOKEN_QUERY,
       &hToken)) {
    printf("OpenProcessToken");
    return FALSE;
  }



  if(fEnable) {
    if (!LookupPrivilegeValue((LPSTR) NULL,
         SE_SHUTDOWN_NAME,
         &ShutdownValue)) {
      printf("LookupPrivilegeValue");
      return FALSE;
    }

    tkp.PrivilegeCount = 1;
    tkp.Privileges[0].Luid = ShutdownValue;
    tkp.Privileges[0].Attributes = SE_PRIVILEGE_ENABLED;

    AdjustTokenPrivileges(hToken,
      FALSE,
      &tkp,
```

```
        sizeof(TOKEN_PRIVILEGES),
        (PTOKEN_PRIVILEGES) NULL,
        (PDWORD) NULL);



      if (GetLastError() != ERROR_SUCCESS) {
        printf("AdjustTokenPrivileges");
        return FALSE;
      }
    }
    else {
      AdjustTokenPrivileges(hToken,
        TRUE,
        (PTOKEN_PRIVILEGES) NULL,
        (DWORD) 0,
        (PTOKEN_PRIVILEGES) NULL,
        (PDWORD) NULL);



      if (GetLastError() != ERROR_SUCCESS) {
        printf("AdjustTokenPrivileges");
        return FALSE;
      }
    }

    return TRUE;
}



char *
GetErrorText( DWORD dwLastError, char *lpszBuf, DWORD dwSize )
{
    DWORD dwRet;
    char *lpszTemp = NULL;

    dwRet = FormatMessage(
FORMAT_MESSAGE_ALLOCATE_BUFFER |

FORMAT_MESSAGE_FROM_SYSTEM |

FORMAT_MESSAGE_ARGUMENT_ARRAY,
                        NULL,
                        dwLastError,
                        LANG_NEUTRAL,
                        ( char * )&lpszTemp,
                        0,
                        NULL );


    if( !dwRet || ( dwSize < dwRet+14 ) )
      lpszBuf[0] = '\0';
    else {
      lpszTemp[strlen(lpszTemp)-2] = '\0';
      sprintf( lpszBuf, "%s (0x%x)", lpszTemp, dwLastError );
    }

    if ( lpszTemp )
      LocalFree((HLOCAL) lpszTemp );

    return lpszBuf;
}
_


#if EXE
int
main( void )
{
  int                                   iCount;
  DWORD                                 dwError;

  printf( "SetRegistryValue: %d\n",
                SetRegistryValue( "SOFTWARE\\Microsoft\\TPCC",
"PATH",
                                        REG_SZ, "bogus_path", sizeof(
"bogus_path" )));

  dwError = GetPerformanceCounterValue( NULL, "HTTP Service", NULL,
NULL, 0,
                                                        "Current
Connections", &iCount );
  if( ERROR_SUCCESS != dwError )
    printf( "Failure: Current Connections: %x\n", dwError );
  else
    printf( "Current Connections: %d\n", iCount );

  return 0;
}
#else
static BOOL GetInstallPath(char *szDllPath)
{
  HKEY      hKey;
  BYTE      szTmp[256];
  char      szKey[256];
  DWORD     size;
  DWORD     sv;
  BOOL      bRc;
  int                   len;
  char      *ptr;
  LONG      status;

  szDllPath[0] =  0;
  bRc = FALSE;
  if ( status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
    "SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Parameters\\Virtual
Roots",
    0, KEY_READ, &hKey) == ERROR_SUCCESS ){
    sv = sizeof(szKey);
    size = sizeof(szTmp);

    if ( RegEnumValue(hKey, 0, szKey, &sv, NULL, NULL, szTmp, &size)
== ERROR_SUCCESS )
    {
      strcpy(szDllPath, szTmp);
      bRc = TRUE;
    }
    RegCloseKey(hKey);
  }
  if ( (ptr = strchr(szDllPath, ',')) )
    *ptr = 0;

  len = strlen(szDllPath);
  if ( szDllPath[len-1] != '\\' )
  {
    szDllPath[len] = '\\';
    szDllPath[len+1] = 0;
  }

  return bRc;
}

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



  switch( ul_reason_for_call )
    {
    case DLL_PROCESS_ATTACH:
      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 );

    break;
   case DLL_THREAD_ATTACH:
    break;
   case DLL_THREAD_DETACH:
    break;
   case DLL_PROCESS_DETACH:
    break;
   }
  return TRUE;
}

static void
Response( EXTENSION_CONTROL_BLOCK *pECB, BOOL success,
char *szStr, ... )
{
  int                           lpbSize;
  int                           iSize;
  char                          szHeader[128];
  char                          szHeader1[128];
  char                          szTmp1[4096];
  char                          szTmp2[4096];
  int                           length;
  va_list                 list;


  va_start( list, szStr );
  length = vsprintf( szTmp1, szStr, list );
  va_end( list );

  length = sprintf( szTmp2, "<BODY> \
                        %s \
                        </BODY>", szTmp1 );

  lpbSize = length + 1;

  if( success ) {
    iSize = sprintf(szHeader, "200 Ok");
    sprintf( szHeader1,
                "Connection: keep-alive\r\n"
                "Content-type: text/html\r\n"
                "Content-length: %d\r\n"
                "\r\n", lpbSize );
  }
  else {
    iSize = sprintf(szHeader, "400 Bad Request");
    sprintf( szHeader1,
                "Content-type: text/html\r\n"
                "Content-length: %d\r\n"
                "\r\n", lpbSize );
  }
  (*pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_SEND_RESPONSE_HEADER,
                                                szHeader, &iSize,
(LPDWORD)szHeader1);
  (*pECB->WriteClient)( pECB->ConnID, szTmp2, &lpbSize, 0 );

  return;
}


BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVer)
{
  if( NULL != pVersionFunc ) {
    return( (*pVersionFunc)( pVer ));
  }
  else {
    pVer->dwExtensionVersion = MAKELONG(HSE_VERSION_MINOR,
HSE_VERSION_MAJOR);
    lstrcpyn(pVer->lpszExtensionDesc, "ISAPI debug.",
HSE_MAX_EXT_DLL_NAME_LEN);
    return TRUE;
  }
}
_

DWORD
MainPage( EXTENSION_CONTROL_BLOCK *pECB )
{
  Response( pECB, TRUE, AdminPageFmt, "NT Admin DLL",
ERROR_SUCCESS,
                "Please select a function.", "", szModName );
  return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}
_


DWORD
ServiceControl( EXTENSION_CONTROL_BLOCK *pECB )
{
  char                          FunctionName[] = "Service Control";

  BOOLEAN                       errflg = FALSE;
  SC_HANDLE                     scmgr;
  SC_HANDLE                     schService =
INVALID_HANDLE_VALUE;
  SERVICE_STATUS     servstat;
  DWORD                         dwLastError;
  char                          service[256];
  char                          *pservice;
  char                          action[256];
  char                          *paction;
  char                          progname[256];
  char                          *pprogname;
  char                          machine[256];
  char                          *pmachine;
  char                          *state;
  char                          arg1[256];
  DWORD                         dwArgc;
  char                          *lpszArgv[10];

  char                          szErrStr[ERRSTRSIZ];

  char                          TmpPage[2048];

  char                          MainPageFmt[] =
        "<form method=GET action=%s> \
        <input type=hidden name=CMD value=ServiceControl> \
        <table> \
                <tr> \
                <th>        </th> \
                <th align=left>Parameter</th> \
                <th>Setting</th> \
                <th align=left>Description</th> \
                </tr> \
                <tr> \
                <td><font color=0000ff>Optional.</font></td> \
                <td>Computer:</td> \
                <td><input type=text name=machine
value=%s></td> \
                <td>Name of machine if other than this web
server.</td> \
                </tr> \
                <tr> \
                <td><font color=ff0000>Required.</font></td> \
                <td>Service:</td> \
                <td><input type=text name=service
value=%s></td> \
                <td>Name of service to perform opeation on.</td>
\
                </tr> \
                <tr> \
                <td><font color=ff0000>Required.</font></td> \
                <td>Action:</td> \
                <td><input type=text name=action
value=%s></td> \
                <td>\"install\", \"start\", \"query\", \"stop\",
\"delete\".</td> \
                </tr> \
                <tr> \
                <td><font color=ffff00>Required if <br> \
                        Action=install.</font></td> \
                <td>Executable:</td> \
                <td><input type=text name=progname
value=%s></td> \
                <td>Fully qualified path of executable.</td> \
                </tr> \
        </table> \
        <p> \
```

```
            <input type=submit value=\"Take Action\"> \
            </form> \
            ";

char                                    ErrorFmt[] = \
            "<table border> \
            <tr> \
            <th>Parameters</th> \
            <tr> \
            <td> \
                        <table> \
                                    <tr> \
                                    <td>Computer:</td> \
                                    <td>%s</td> \
                                    </tr> \
                                    <tr> \
                                    <td>Service:</td> \
                                    <td>%s</td> \
                                    </tr> \
                                    <tr> \
                                    <td>Action:</td> \
                                    <td>%s</td> \
                                    </tr> \
                                    <tr> \
                                    <td align=right>Executable:</td> \
                                    <td align=left>%s</td> \
                                    </tr> \
                                    <tr> \
                                    <td>State:</td> \
                                    <td>%s</td> \
                                    </tr> \
                        </table> \
            </td> \
            </tr> \
            </table> \
            ";


 pservice = service;
 service[0] = '\0';
 paction = action;
 action[0] = '\0';
 pprogname = progname;
 progname[0] = '\0';
 pmachine = machine;
 machine[0] = '\0';


 if( !GetKeyValue( pECB->lpszQueryString, "service", pservice )) {
  pservice = NULL;
 }

 if( !GetKeyValue( pECB->lpszQueryString, "action", paction )) {
  paction = NULL;
 }
 else if( 0 == strcmp( action, "install" )) {

  if( !GetKeyValue( pECB->lpszQueryString, "progname", pprogname )) {
   pprogname = NULL;
  }
 }

 if( !GetKeyValue( pECB->lpszQueryString, "machine", pmachine )) {
  pmachine = NULL;
 }


 if( NULL == pservice || NULL == paction ||
    (0 == strcmp( action, "install") && NULL == pprogname)) {
  sprintf( TmpPage, MainPageFmt, szModName,
                machine, service, action, progname );
  Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                "Enter all required data.", TmpPage, szModName );
  return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
 }

 scmgr = OpenSCManager( pmachine, NULL,
SC_MANAGER_ALL_ACCESS );
 if( NULL == scmgr ) {
  dwLastError = GetLastError( );
  sprintf( TmpPage, ErrorFmt, machine, service, action, progname, "" );
  Response( pECB, FALSE, AdminPageFmt, FunctionName, dwLastError,
                GetErrorText( dwLastError, szErrStr, ERRSTRSIZ ),
                TmpPage, szModName );
  return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
 }

 dwLastError = ERROR_SUCCESS;
 if( 0 == strcmp( action, "install" )) {
  schService = CreateService( scmgr, service, service,

            SERVICE_ALL_ACCESS,

            SERVICE_WIN32_OWN_PROCESS,

            SERVICE_AUTO_START, SERVICE_ERROR_NORMAL,
                                    progname, NULL,
NULL, NULL, NULL, NULL );
  if( schService == NULL ) {
   dwLastError = GetLastError( );
  }
 }
 else if( 0 == strcmp( action, "start" )) {
  schService = OpenService (scmgr, service, SERVICE_ALL_ACCESS);
  if (schService == NULL) {
   dwLastError = GetLastError( );
  }

  if( !GetKeyValue( pECB->lpszQueryString, "arg1", arg1 )) {
   dwArgc = 0;
   lpszArgv[0] = NULL;
  }
  else {
   dwArgc = 1;
   lpszArgv[0] = arg1;
  }

  if (! StartService (schService, dwArgc, lpszArgv)) {
   dwLastError = GetLastError( );
  }

  Sleep( 5000 );
 }
 else if( 0 == strcmp( action, "stop" )) {
  schService = OpenService (scmgr, service, SERVICE_ALL_ACCESS);
  if (schService == NULL) {
   dwLastError = GetLastError( );
  }

  if (! ControlService (schService, SERVICE_CONTROL_STOP,
&servstat)) {
   dwLastError = GetLastError( );
  }

  Sleep( 5000 );
 }
 else if( 0 == strcmp( action, "query" )) {
  schService = OpenService (scmgr, service, SERVICE_ALL_ACCESS);
  if (schService == NULL) {
   dwLastError = GetLastError( );
  }
 }
 else if( 0 == strcmp( action, "delete" )) {
  schService = OpenService (scmgr, service, SERVICE_ALL_ACCESS);
  if (schService == NULL) {
   dwLastError = GetLastError( );
  }

  if (! DeleteService(schService)) {
   dwLastError = GetLastError( );
  }
 }
 else {
  dwLastError = ERROR_INVALID_FUNCTION;
 }

 if( ERROR_SUCCESS == dwLastError )
 {
```

```
  if( ! QueryServiceStatus( schService, &servstat )) {
    dwLastError = GetLastError( );
  }
 }

 if( ERROR_SUCCESS == dwLastError )
 {
  if( SERVICE_STOPPED  == servstat.dwCurrentState )
    state = SERVICE_CONTROL_STATE_STOPPED;
  else if( SERVICE_START_PENDING  == servstat.dwCurrentState )
    state = SERVICE_CONTROL_STATE_START_PENDING;
  else if( SERVICE_STOP_PENDING  == servstat.dwCurrentState )
    state = SERVICE_CONTROL_STATE_STOP_PENDING;
  else if( SERVICE_RUNNING  == servstat.dwCurrentState )
    state = SERVICE_CONTROL_STATE_RUNNING;
  else if( SERVICE_CONTINUE_PENDING  == servstat.dwCurrentState )
    state = SERVICE_CONTROL_STATE_CONTINUE_PENDING;
  else if( SERVICE_PAUSE_PENDING  == servstat.dwCurrentState )
    state = SERVICE_CONTROL_STATE_PAUSE_PENDING;
  else if( SERVICE_PAUSED == servstat.dwCurrentState )
    state = SERVICE_CONTROL_STATE_PAUSED;
  sprintf( TmpPage, ServiceControlSuccessFmt,
                machine, service, action, progname, state );
  Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                "Execution Complete", TmpPage, szModName );
 }
 else
 {
  sprintf( TmpPage, ErrorFmt, machine, service, action, progname, "" );
  Response( pECB, FALSE, AdminPageFmt, FunctionName, dwLastError,
                GetErrorText( dwLastError, szErrStr, ERRSTRSIZ ),
                TmpPage, szModName );
 }

 if (INVALID_HANDLE_VALUE != schService)
  CloseServiceHandle (schService);
 CloseServiceHandle (scmgr);

 return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}
_

DWORD
GetPerformanceCounter( EXTENSION_CONTROL_BLOCK *pECB )
{
 char                    FunctionName[] = "Get Performance
Counter";

 char                    machine[256];
 char                    *pmachine;
 char                    object[256];
 char                    *pobject;
 char                    instance[256];
 char                    *pinstance;
 char                    parent[256];
 char                    *pparent;
 char                    counter[256];
 char                    *pcounter;
 char                    szindex[256];
 DWORD                   index;
 int                     count;
 DWORD                   status;
 DWORD                   size_dw = sizeof( DWORD );

 char                    TmpPage[2048];

 char                    MainPageFmt[] = \
        "<form method=GET action=%s> \
        <input type=hidden name=CMD
value=GetPerformanceCounter> \
        <table> \
                <tr> \
                <th>    </th> \
                <th align=left>Parameter</th> \
                <th>Setting</th> \
                <th align=left>Description</th> \
                </tr> \
                <tr> \
                <td><font color=0000ff>Optional.</font></td> \
                <td>Computer:</td> \
                <td><input type=text name=machine
value=%s></td> \
                <td>Name of machine if other than this web
server.</td> \
                </tr> \
                <tr> \
                <td><font color=ff0000>Required.</font></td> \
                <td>Object:</td> \
                <td><input type=text name=object
value=%s></td> \
                <td>Name of Object (e.g. Processor, HTTP
Service, etc.).</td> \
                </tr> \
                <tr> \
                <td><font color=ff0000>Required.</font></td> \
                <td>Counter:</td> \
                <td><input type=text name=counter
value=%s></td> \
                <td>Performance counter to get.</td> \
                </tr> \
                <tr> \
                <td><font color=0000ff>Optional.</font></td> \
                <td>Instance:</td> \
                <td><input type=text name=instance
value=%s></td> \
                <td>Particular instance of Object (e.g. CPU 1 vs.
CPU 3).</td> \
                </tr> \
                <tr> \
                <td><font color=0000ff>Optional</font></td> \
                <td>Parent:</td> \
                <td><input type=text name=parent
value=%s></td> \
                <td>Parent of instance.</td> \
                </tr> \
                <tr> \
                <td><font color=0000ff>Optional.</font></td> \
                <td>Index:</td> \
                <td><input type=text name=index
value=%s></td> \
                <td>???.</td> \
                </tr> \
        </table> \
        <p> \
        <input type=submit value=\"Collect Counter Value\"> \
        </form> \
        ";


 char                    SuccessFmt[] = "Counter Value: %d";

 char                    ErrorFmt[] = \
        "<table border> \
        <tr> \
        <th>Parameters \
        <tr> \
        <td> \
                <table> \
                        <tr> \
                        <td>Computer:</td> \
                        <td>%s</td> \
                        </tr> \
                        <tr> \
                        <td>Object:</td> \
                        <td>%s</td> \
                        </tr> \
                        <tr> \
                        <td>Counter:</td> \
                        <td>%s</td> \
                        </tr> \
                        <tr> \
                        <td>Instance:</td> \
                        <td>%s</td> \
                        </tr> \
                        <tr> \
                        <td>Parent:</td> \
                        <td>%s</td> \
                        </tr> \
                        <tr> \
```

```
                                        <td>Index:</td> \
                                        <td>%s</td> \
                                        </tr> \
                                </table> \
                </td> \
                </tr> \
                </table> \
                ";


  machine[0] = '\0';
  pmachine = machine;

  object[0] = '\0';
  pobject = object;

  instance[0] = '\0';
  pinstance = instance;

  parent[0] = '\0';
  pparent = parent;

  counter[0] = '\0';
  pcounter = counter;

  szindex[0] = '\0';


  pmachine = machine;
  if( !GetKeyValue( pECB->lpszQueryString, "machine", pmachine )) {
    pmachine = NULL;
  }

  if( !GetKeyValue( pECB->lpszQueryString, "object", object )) {
    pobject = NULL;
  }

  pinstance = instance;
  if( !GetKeyValue( pECB->lpszQueryString, "instance", pinstance )) {
    pinstance = NULL;
  }

  pparent = parent;
  if( !GetKeyValue( pECB->lpszQueryString, "parent", pparent )) {
    pparent = NULL;
  }

  if( !GetKeyValue( pECB->lpszQueryString, "index", szindex )) {
    index = (DWORD)-1;
  }
  else {
    index = atoi( szindex );
  }

  if( !GetKeyValue( pECB->lpszQueryString, "counter", counter )) {
    pcounter=NULL;
  }


  if( NULL == pobject || NULL == pcounter ) {
    sprintf( TmpPage, MainPageFmt, szModName,
             machine, object, counter, instance, parent, szindex );
    Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
              "Enter all required data.", TmpPage, szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }

  status = GetPerformanceCounterValue( pmachine, object, pinstance,
pparent,
                                       index, counter,
&count );
  if( ERROR_SUCCESS != status ) {
    sprintf( TmpPage, ErrorFmt, machine, object, counter, instance, parent,
             szindex );
    Response( pECB, FALSE, AdminPageFmt, FunctionName, status,
              "Failed to get performance counter.", TmpPage,
szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }

  sprintf( TmpPage, SuccessFmt, count );
  Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
            "Got performance counter value.", TmpPage, szModName );
  return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}
_

DWORD
ShutdownSystem( EXTENSION_CONTROL_BLOCK *pECB )
{
  char                  FunctionName[] = "Shutdown
System";

  DWORD                 dwLastError;

  char                  Machine[256];
  char                  *pMachine;

  char                  Msg[256];
  char                  *pMsg;

  char                  Delay[256];
  char                  *pDelay;
  int                   iDelay;

  char                  Force[256];
  char                  *pForce;
  BOOL                  bForce;

  char                  Reboot[256];
  char                  *pReboot;
  BOOL                  bReboot;

  DWORD                 Status;

  char                  szErrStr[ERRSTRSIZ];

  char                  TmpPage[2048];

  char                  MainPageFmt[] = \
        "<form method=GET action=%s> \
        <input type=hidden name=CMD value=ShutdownSystem> \
        <table> \
                <tr> \
                <th>    </th> \
                <th align=left>Parameter</th> \
                <th>Setting</th> \
                <th align=left>Description</th> \
                </tr> \
                <td><font color=0000ff>Optional.</font></td> \
                <td>Computer:</td> \
                <td><input type=text name=machine
value=%s></td> \
                <td>Name of computer if other than this web
server.</td> \
                </tr> \
                <tr> \
                <td><font color=0000ff>Optional.</font></td> \
                <td>Message:</td> \
                <td><input type=text name=message
value=%s></td> \
                <td>Message to display in shutdown dialog
box.</td> \
                </tr> \
                <td><font color=0000ff>Optional.</font></td> \
                <td>Delay Time:</td> \
                <td><input align=right type=text name=delay
value=%d></td> \
                <td>Maximum delay time, in seconds, until
shutdown.</td> \
                </tr> \
        </table> \
        <p> \
        <table> \
                <tr> \
                <th>Yes</th> \
                <th>No</th> \
```

```
                        <th></th> \
                        <tr> \
                        <td><input type=radio name=force value=1></td>
\
                        <td><input type=radio name=force checked
value=0></td> \
                        <td>Force a system shutdown inspite of unsaved
application changes.</td> \
                        </tr> \
                        <tr> \
                        <td><input type=radio name=reboot checked
value=1></td> \
                        <td><input type=radio name=reboot
value=0></td> \
                        <td>Reboot the machine immediately after the
system shutdown.</td> \
                        </tr> \
                </table> \
                <p> \
                <input type=submit value=\"Shut the system down\"> \
                </form> \
                ";

 char                                   SuccessFmt[] = \
                "<table border> \
                <tr> \
                <th>Parameters</th> \
                <tr> \
                <td> \
                        <table> \
                                <tr> \
                                <td>Computer:</td> \
                                <td>%s</td> \
                                </tr> \
                                <tr> \
                                <td>Message:</td> \
                                <td>%s</td> \
                                </tr> \
                                <tr> \
                                <td>Delay:</td> \
                                <td>%d</td> \
                                </tr> \
                                <tr> \
                                <td>Force:</td> \
                                <td>%d</td> \
                                </tr> \
                                <tr> \
                                <td>Reboot:</td> \
                                <td>%d</td> \
                                </tr> \
                        </table> \
                </td> \
                </tr> \
                </table> \
                ";

 char                                   ErrorFmt[] = \
                "<table border> \
                <tr> \
                <th>Parameters</th> \
                <tr> \
                <td> \
                        <table> \
                                <tr> \
                                <td>Computer:</td> \
                                <td>%s</td> \
                                </tr> \
                                <tr> \
                                <td>Message:</td> \
                                <td>%s</td> \
                                </tr> \
                                <tr> \
                                <td>Delay:</td> \
                                <td>%d</td> \
                                </tr> \
                                <tr> \
                                <td>Force:</td> \
                                <td>%d</td> \
                                </tr> \
                                <tr> \
                                <td>Reboot:</td> \
                                <td>%d</td> \
                                </tr> \
                        </table> \
                </td> \
                </tr> \
                </table> \
                ";


 Machine[0] = '\0';
 pMachine = Machine;

 Msg[0] = '\0';
 pMsg = Msg;

 Delay[0] = '\0';
 pDelay = Delay;
 iDelay = 0;

 Force[0] = '\0';
 pForce = Force;
 bForce = TRUE;


 Reboot[0] = '\0';
 pReboot = Reboot;
 bReboot = FALSE;


 if( !GetKeyValue( pECB->lpszQueryString, "machine", pMachine )) {
   pMachine = NULL;
 }

 if( !GetKeyValue( pECB->lpszQueryString, "message", pMsg )) {
   pMsg = NULL;
 }

 if( !GetKeyValue( pECB->lpszQueryString, "delay", pDelay )) {
   pDelay = NULL;
 }
 else {
   iDelay = atoi( pDelay );
 }

 if( !GetKeyValue( pECB->lpszQueryString, "force", pForce )) {
   pForce = NULL;
   bForce = FALSE;
 }
 else {
   bForce = atoi( pForce );
 }

 if( !GetKeyValue( pECB->lpszQueryString, "reboot", pReboot )) {
   pReboot = NULL;
   bReboot = FALSE;
 }
 else {
   bReboot = atoi( pReboot );
 }


  if( NULL == pMachine && NULL == pMsg && NULL == pDelay &&
NULL == pForce &&
      NULL == pReboot ) {
    sprintf( TmpPage, MainPageFmt, szModName,
                    Machine, Msg, iDelay, bForce, bReboot );
    Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                    "Enter all required data.", TmpPage, szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }


  AssertShutdown(TRUE);

  Status = InitiateSystemShutdown(pMachine, pMsg, iDelay, bForce,
bReboot);

  if( 0 == Status ) {
```

```
    dwLastError = GetLastError( );
    sprintf( TmpPage, ErrorFmt, Machine, Msg, iDelay, bForce, bReboot );
    Response( pECB, FALSE, AdminPageFmt, FunctionName, dwLastError,
                GetErrorText( dwLastError, szErrStr, ERRSTRSIZ ),
                TmpPage, szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }
  sprintf( TmpPage, SuccessFmt, Machine, Msg, iDelay, bForce, bReboot );
  Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
            "System shutting down.", TmpPage, szModName );
  return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}
_

DWORD
GetRegistry( EXTENSION_CONTROL_BLOCK *pECB )
{
  char                          FunctionName[] = "Get Registry";

  char                          key[256];
  char                          *pkey;
  char                          value[256];
  char                          *pvalue;
  char                          type[256];
  char                          *ptype;
  char                          data[4096];
  char                          multistr[8192];
  char                          *pdata;
  DWORD                         dwTmp;
  LONG                          status;
  DWORD                         size_dw = sizeof( DWORD );
  DWORD                         len;
  DWORD                         len2;

  char                          TmpPage[2048];

  char                          MainPageFmt[] = \
        "<form method=GET action=%s> \
        <input type=hidden name=CMD value=GetRegistry> \
        <table> \
                <tr> \
                <th>    </th> \
                <th align=left>Parameter</th> \
                <th>Setting</th> \
                <th align=left>Description</th> \
                </tr> \
                <tr> \
                <td><font color=ff0000>Required.</font></td> \
                <td>Key:</td> \
                <td><input type=text name=key value=%s></td> \
                <td>Name of Registry Key.</td> \
                </tr> \
                <tr> \
                <td><font color=ff0000>Required.</font></td> \
                <td>Value:</td> \
                <td><input type=text name=value
value=%s></td> \
                <td>Name of Registry Value.</td> \
                </tr> \
        </table> \
        <p> \
        <input type=submit value=\"Get Registry Value\"> \
        </form> \
        ";

  char                          ErrorFmt[] = \
        "<table border>"
        "<tr>"
        "<th>Parameters</th>"
        "<tr>"
        "<td>"
                "<table>"

        "<tr><td>Key:</td><td>%s</td></tr>"

        "<tr><td>Value:</td><td>%s</td></tr>"

        "<tr><td>Type:</td><td>%s</td></tr>"
                                                "<tr><td
valign=top>Data:</td><td>%s</td></tr>"
                                "</table>"
        "</td>"
        "</tr>"
        "</table>"
        ;



  key[0] = '\0';
  pkey = key;

  value[0] = '\0';
  pvalue = value;

  type[0] = '\0';
  ptype = type;

  data[0] = '\0';
  pdata = data;


  if( !GetKeyValue( pECB->lpszQueryString, "key", key )) {
    pkey = NULL;
  }

  if( !GetKeyValue( pECB->lpszQueryString, "value", value )) {
    pvalue = NULL;
  }


  if( NULL == pkey || NULL == pvalue ) {
    sprintf( TmpPage, MainPageFmt, szModName, key, value, type, data );
    Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
            "Enter all required data.", TmpPage, szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }


  status = GetRegistryValue( key, value, &dwTmp, NULL, &len );
  if( ERROR_FILE_NOT_FOUND == status ) {
    sprintf( TmpPage, GetRegistrySuccessFmt, key, value, type, data );
    Response( pECB, TRUE, AdminPageFmt, FunctionName, status,
            "Registry value is not set.", TmpPage, szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }
  else if( ERROR_SUCCESS != status ) {
    sprintf( TmpPage, ErrorFmt, key, value, type, data );
    Response( pECB, FALSE, AdminPageFmt, FunctionName, status,
            "Failed to get registry value.", TmpPage, szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }

  if( REG_DWORD == dwTmp ) {
    status = GetRegistryValue( key, value, NULL,
                                        (LPBYTE)&dwTmp, &size_dw
);
    if( ERROR_SUCCESS != status ) {
      sprintf( TmpPage, ErrorFmt, key, value, type, data );
      Response( pECB, FALSE, AdminPageFmt, FunctionName, status,
                        "Failed to get registry value.", TmpPage,
szModName );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
    else {
      sprintf( data, "%u", dwTmp );
      sprintf( TmpPage, GetRegistrySuccessFmt, key, value, "dword", data );
      Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                        "Registry value has been gotten.", TmpPage,
szModName );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }
  else if( REG_MULTI_SZ == dwTmp ) {
    len = sizeof( data ) - 1;
    status = GetRegistryValue( key, value, NULL, data, &len );
    if( ERROR_SUCCESS != status ) {
```

```
    sprintf( TmpPage, ErrorFmt, key, value, type, data );
    Response( pECB, FALSE, AdminPageFmt, FunctionName, status,
                        "Failed to get registry value.", TmpPage,
szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
   }
   else {
    data[len] = '\0';
    strcpy( multistr, data );
    len2 = strlen( data ) + 1;
    while( '\0' != data[len2] ) {
            strcat( multistr, "<br>" );
            strcat( multistr, &data[len2] );
            len2 += strlen( &data[len2] ) + 1;
    }
    sprintf( TmpPage, GetRegistrySuccessFmt, key, value,
                "multi_string", multistr );
    Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                        "Registry value has been gotten.", TmpPage,
szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
   }
  }
  else if( REG_SZ == dwTmp ) {
   len = sizeof( data ) - 1;
   status = GetRegistryValue( key, value, NULL, data, &len );
   if( ERROR_SUCCESS != status ) {
    sprintf( TmpPage, ErrorFmt, key, value, type, data );
    Response( pECB, FALSE, AdminPageFmt, FunctionName, status,
                        "Failed to get registry value.", TmpPage,
szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
   }
   else {
    data[len] = '\0';
    sprintf( TmpPage, GetRegistrySuccessFmt, key, value, "string", data );
    Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                        "Registry value has been gotten.", TmpPage,
szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
   }
  }
  else {
   sprintf( TmpPage, ErrorFmt, key, value, type, data );
   Response( pECB, FALSE, AdminPageFmt, FunctionName, -1,
"Unknown type.",
                TmpPage, szModName );
   return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }
}
_

DWORD
SetRegistry( EXTENSION_CONTROL_BLOCK *pECB )
{
  char                          FunctionName[] = "Set Registry";

  char                          key[256];
  char                          *pkey;
  char                          value[256];
  char                          *pvalue;
  char                          type[256];
  char                          *ptype;
  char                          data[4096];
  char                          multistr[8192];
  char                          *pdata;
  DWORD                         dwTmp;
  LONG                          status;
  DWORD                         size_dw = sizeof( DWORD );
  char                          TmpPage[2048];
  DWORD                         len;
  DWORD                         len2;
  DWORD                         idx1;
  DWORD                         idx2;

  char                          MainPageFmt[] = \
            "<form method=GET action=%s> \
            <input type=hidden name=CMD value=SetRegistry> \
            <table> \
                        <tr> \
                        <th>        </th> \
                        <th align=left>Parameter</th> \
                        <th>Setting</th> \
                        <th align=left>Description</th> \
                        </tr> \
                        <tr> \
                        <td><font color=ff0000>Required.</font></td> \
                        <td>Key:</td> \
                        <td><input type=text name=key value=%s></td> \
                        <td>Name of Registry Key.</td> \
                        </tr> \
                        <tr> \
                        <td><font color=ff0000>Required.</font></td> \
                        <td>Value:</td> \
                        <td><input type=text name=value
value=%s></td> \
                        <td>Name of Registry Value to be set.</td> \
                        </tr> \
                        <tr> \
                        <td><font color=ff0000>Required.</font></td> \
                        <td>Type:</td> \
                        <td><input type=text name=type value=%s></td>
\
                        <td>Type of Value to be set (string, dword).</td> \
                        </tr> \
                        <tr> \
                        <td><font color=ff0000>Required.</font></td> \
                        <td>Data:</td> \
                        <td> \
                                <textarea name=data cols=50 rows=3
value=%s> \
                                </textarea> \
                        </td> \
                        <td>New setting for Value.</td> \
                        </tr> \
            </table> \
            <p> \
            <input type=submit value=\"Set Registry Value\"> \
            </form> \
            ";

  char                          SuccessFmt[] = \
            "<table border> \
            <tr> \
            <th>Parameters</th> \
            <tr> \
            <td> \
                        <table> \
                                <tr> \
                                <td>Key:</td> \
                                <td>%s</td> \
                                </tr> \
                                <tr> \
                                <td>Value:</td> \
                                <td>%s</td> \
                                </tr> \
                                <tr> \
                                <td>Type:</td> \
                                <td>%s</td> \
                                </tr> \
                                <tr> \
                                <td valign=top>Data:</td> \
                                <td>%s</td> \
                                </tr> \
                        </table> \
            </td> \
            </tr> \
            </table> \
            ";

  char                          ErrorFmt[] = \
            "<table border> \
            <tr> \
            <th>Parameters</th> \
            <tr> \
            <td> \
                        <table> \
                                <tr> \
```

```
                                        <td>Key:</td> \
                                        <td>%s</td> \
                                        </tr> \
                                        <tr> \
                                        <td>Value:</td> \
                                        <td>%s</td> \
                                        </tr> \
                                        <tr> \
                                        <td>Type:</td> \
                                        <td>%s</td> \
                                        </tr> \
                                        <tr> \
                                        <td valign=top>Data:</td> \
                                        <td>%s</td> \
                                        </tr> \
                                </table> \
                </td> \
                </tr> \
                </table> \
                ";


  key[0] = '\0';
  pkey = key;

  value[0] = '\0';
  pvalue = value;

  type[0] = '\0';
  ptype = type;

  data[0] = '\0';
  pdata = data;


  if( !GetKeyValue( pECB->lpszQueryString, "key", key )) {
    pkey = NULL;
  }

  if( !GetKeyValue( pECB->lpszQueryString, "value", value )) {
    pvalue = NULL;
  }

  if( !GetKeyValue( pECB->lpszQueryString, "type", type )) {
    ptype = NULL;
  }

  if( !GetKeyValue( pECB->lpszQueryString, "data", data )) {
    pdata = NULL;
  }


  if( NULL == pkey || NULL == pvalue || NULL == ptype ||
      (( NULL == pdata ) &&
       ( 0 != strncmp( "multi_string", type, 12 )) &&
       ( 0 != strncmp( "string", type, 6 )))) {
    sprintf( TmpPage, MainPageFmt, szModName, key, value, type, data );
    Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                  "Enter all required data.", TmpPage, szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  }

  if( 0 == strncmp( "dword", type, 5 )) {
    dwTmp = atoi( data );
    status = SetRegistryValue( key, value, REG_DWORD,
                                        (LPBYTE)&dwTmp, size_dw );
    if( ERROR_SUCCESS != status ) {
      sprintf( TmpPage, ErrorFmt, key, value, type, data );
      Response( pECB, FALSE, AdminPageFmt, FunctionName, status,
                        "Failed to set registry value.", TmpPage,
szModName );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
    else {
      sprintf( TmpPage, SuccessFmt, key, value, type, data );
      Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                        "Registry value has been set.", TmpPage,
szModName );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }
  else if( 0 == strncmp( "multi_string", type, 12 )) {
    len = strlen( data );
    if( 0 == len ) {


      len = 1;
    }

    idx1 = idx2 = 0;
    while( idx1 < len ) {
      if( '\r' == data[idx1] ) {
                data[idx2++] = '\0';
                idx1 += 2;
      }
      else {
                data[idx2++] = data[idx1++];
      }
    }
    data[idx2++] = '\0';
    data[idx2++] = '\0';
    len = idx2;
    status = SetRegistryValue( key, value, REG_MULTI_SZ, data, len );
    if( ERROR_SUCCESS != status ) {
      sprintf( TmpPage, ErrorFmt, key, value, type, data );
      Response( pECB, FALSE, AdminPageFmt, FunctionName, status,
                        "Failed to set registry value.", TmpPage,
szModName );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
    else {
      strcpy( multistr, data );
      len2 = strlen( data ) + 1;
      while( '\0' != data[len2] ) {
                strcat( multistr, "<br>" );
                strcat( multistr, &data[len2] );
                len2 += strlen( &data[len2] ) + 1;
      }
      sprintf( TmpPage, SuccessFmt, key, value, type, multistr );
      Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                        "Registry value has been set.", TmpPage,
szModName );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }
  else if( 0 == strncmp( "string", type, 6 )) {
    len = strlen( data );
    if( 0 == len ) {


      len = 1;
    }
    status = SetRegistryValue( key, value, REG_SZ, data, len );
    if( ERROR_SUCCESS != status ) {
      sprintf( TmpPage, ErrorFmt, key, value, type, data );
      Response( pECB, FALSE, AdminPageFmt, FunctionName, status,
                        "Failed to set registry value.", TmpPage,
szModName );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
    else {
      sprintf( TmpPage, SuccessFmt, key, value, type, data );
      Response( pECB, TRUE, AdminPageFmt, FunctionName,
ERROR_SUCCESS,
                        "Registry value has been set.", TmpPage,
szModName );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }
  else {
    sprintf( TmpPage, ErrorFmt, key, value, type, data );
    Response( pECB, FALSE, AdminPageFmt, FunctionName, -1,
"Unknown type.",
                  TmpPage, szModName );
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
```

```
  }
}
_
static void
SysErrResponse( EXTENSION_CONTROL_BLOCK *pECB, char
*szErrStr )
{
#define ERRSTRSIZ 256
  char                                  szTmp[ERRSTRSIZ];

  Response( pECB, FALSE, szErrStr,
                GetErrorText( GetLastError( ), szTmp, ERRSTRSIZ ));
}
_

DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK
*pECB)
{
#define MAX_CMD_LEN 30
  char Cmd[MAX_CMD_LEN];

  if( GetKeyValue( pECB->lpszQueryString, "CMD", Cmd )) {

    if( 0 == strncmp( Cmd, "GetPerformanceCounter",
                              strlen( "GetPerformanceCounter" ))) {
      return( GetPerformanceCounter( pECB ));
    }
    else if( 0 == strncmp( Cmd, "GetRegistry",
                                          strlen( "GetRegistry" ))) {
      return( GetRegistry( pECB ));
    }
    else if( 0 == strncmp( Cmd, "SetRegistry",
                                          strlen( "SetRegistry" ))) {
      return( SetRegistry( pECB ));
    }
    else if( 0 == strncmp( Cmd, "ServiceControl",
                                          strlen( "ServiceControl" ))) {
      return( ServiceControl( pECB ));
    }
    else if( 0 == strncmp( Cmd, "ShutdownSystem",
                                          strlen( "ShutdownSystem" ))) {
      return( ShutdownSystem( pECB ));
    }
    else {
      Response( pECB, FALSE, ERROR_NOT_RECOGNIZED, Cmd );
      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }
  else {

    return( MainPage( pECB ));
  }
}
#endif

DWORD
GetPerformanceCounterValue( char *machine, char *object, char *instance,
                                          char *parent, DWORD index, char
*counter,
                                          int *count )
{
#ifdef USE_REGISTRY_DIRECTLY_TO_GET_PERF_DATA
  HKEY                          hKey =
HKEY_PERFORMANCE_DATA;
  LONG                          lError;
  DWORD                         dwType;
  DWORD                         dwPerfDataLen, *pdwPerfDataLen =
&dwPerfDataLen;
  PPERF_DATA_BLOCK              pPerfData = NULL;
#define EXTRA_MEM   0x1000
  int                           iSize;
  WCHAR                         szValue[] = L"1880";
  WCHAR                         *lpszValue = szValue;


  iSize = sizeof( PERF_DATA_BLOCK ) + EXTRA_MEM;
  pPerfData = malloc( iSize );
  *pdwPerfDataLen = iSize;

  while( 1 ) {
    lError = RegQueryValueExW( hKey, lpszValue, NULL, &dwType,
                                          ( LPSTR )pPerfData,
pdwPerfDataLen );
    if( ERROR_MORE_DATA == lError ) {
      iSize += EXTRA_MEM;
      pPerfData = realloc( pPerfData, iSize );
      if( NULL == pPerfData ) {
                return ERROR_NOT_ENOUGH_MEMORY;
      }
    }
    else if( ERROR_SUCCESS != lError ) {
      return lError;
    }
    break;
  }

  return ERROR_SUCCESS;
#else
  HQUERY                hQuery;
  HCOUNTER                      hCounter;
  PDH_STATUS                    pdhStatus;
  PDH_FMT_COUNTERVALUE          fmtValue;
  DWORD                         ctrType;
  PDH_COUNTER_PATH_ELEMENTS elements;
#define CPATHBUFSIZ 4096
  char                          szFullPath[CPATHBUFSIZ];
  DWORD                         dwBufferSize = CPATHBUFSIZ;


  pdhStatus = PdhOpenQuery (0, 0, &hQuery);
  assert (pdhStatus == ERROR_SUCCESS);

  elements.szMachineName = machine;
  elements.szObjectName = object;
  elements.szInstanceName = instance;
  elements.szParentInstance = parent;
  elements.dwInstanceIndex = index;
  elements.szCounterName = counter;

  pdhStatus = PdhMakeCounterPath( &elements, szFullPath,
&dwBufferSize, 0 );
  if( pdhStatus != ERROR_SUCCESS ) {
    return pdhStatus;
  }

  pdhStatus = PdhAddCounter( hQuery, szFullPath, 0, &hCounter );
  if( pdhStatus != ERROR_SUCCESS ) {
    return pdhStatus;
  }


  pdhStatus = PdhCollectQueryData (hQuery);
  if( pdhStatus != ERROR_SUCCESS ) {
    return pdhStatus;
  }
  pdhStatus = PdhCollectQueryData (hQuery);
  if( pdhStatus != ERROR_SUCCESS ) {
    return pdhStatus;
  }
  pdhStatus = PdhGetFormattedCounterValue( hCounter,
PDH_FMT_LONG,
                                                                  &ctrType,
&fmtValue );
  if( pdhStatus != ERROR_SUCCESS ) {
    return pdhStatus;
  }


  if( ERROR_SUCCESS != fmtValue.CStatus ) {
    return fmtValue.CStatus;
  }

  *count = (int)fmtValue.longValue;


  pdhStatus = PdhCloseQuery (hQuery);
  if( pdhStatus != ERROR_SUCCESS ) {
    return pdhStatus;
  }
```

```
  return pdhStatus;
#endif
}
_

LONG
GetRegistryValue( char *key, char *value, DWORD *ptype,
                                LPBYTE data, DWORD *psize )
{
  LONG      status;
  HKEY      hKey;

  status = RegOpenKeyEx( HKEY_LOCAL_MACHINE, key, 0,
KEY_QUERY_VALUE, &hKey );
  if( ERROR_SUCCESS != status )
    return status;

  status = RegQueryValueEx( hKey, value, 0, ptype, data, psize );
  if( ERROR_SUCCESS != status )
    return status;

  RegCloseKey( hKey );

}
_

LONG
SetRegistryValue( char *key, char *value, DWORD type, LPBYTE data,
DWORD size )
{
  LONG      status;
  DWORD   dwDisposition;
  HKEY      hKey;

  status = RegCreateKeyEx(HKEY_LOCAL_MACHINE, key,
                                        0, NULL,
REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS,
                                        NULL, &hKey, &dwDisposition);
  if( ERROR_SUCCESS != status )
    return status;

  status = RegSetValueEx( hKey, value, 0, type, data, size );
  if( ERROR_SUCCESS != status )
    return status;

  RegCloseKey( hKey );

}
_

BOOL
GetKeyValue( char *query, char *key, char *value )
{
  char                          *pszTmp1;
  char                          *pszTmp2;
  int                           len;
  int                           ii;
  int                           jj;
  int                           num;


  if( NULL == ( pszTmp1 = strstr( query, key )))
    return FALSE;


  pszTmp1 += strlen( key ) + 1;
  if( NULL == ( pszTmp2 = strchr( pszTmp1, '&' ))) {


    strcpy( value, pszTmp1 );
  }
  else {

    len = pszTmp2 - pszTmp1;
    memcpy( value, pszTmp1, len );
    value[len] = '\0';
  }


  if( '\0' == *value ) {
    return FALSE;
  }


 for( ii = 0, jj = 0; '\0' != value[ii]; ii++, jj++ ) {
   if( '%' == value[ii] ) {


     ii++;
     num = 0;


     if( '0' <= value[ii] && '9' >= value[ii] ) {
               num += ( value[ii++] - '0' );
     }
     else if( 'a' <= value[ii] && 'f' >= value[ii] ) {
               num += 10 + ( value[ii++] - 'a' );
     }
     else if('A' <= value[ii] && 'F' >= value[ii] ) {
               num += 10 + ( value[ii++] - 'A' );
     }

     num *= 0x10;


     if( '0' <= value[ii] && '9' >= value[ii] ) {
               num += ( value[ii] - '0' );
     }
     else if( 'a' <= value[ii] && 'f' >= value[ii] ) {
               num += 10 + ( value[ii] - 'a' );
     }
     else if('A' <= value[ii] && 'F' >= value[ii] ) {
               num += 10 + ( value[ii] - 'A' );
     }

     value[jj] = num;

   } else if( '+' == value[ii] ) {

     value[jj] = ' ';
   } else {

     value[jj] = value[ii];
   }
 }

 value[jj] = '\0';

 return TRUE;
}
```

aui_web_srv.h

```
#ifndef AUI_WEB_SRV_H
#define AUI_WEB_SRV_H
/*+************************************************************
**************
 *                                                            *
 *  COPYRIGHT (c) 1997, 2000 BY                               *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
```

```
 *  CORPORATION.                                        *
 *                                                      *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.             *
 *                                                      *
 *                                                      *

**********************************************************
************-*/
/*+            FILE:                   AUI_WEB_SRV.H
 *                                                  Copyright Digital, 1996-
1999
 *
 *            PURPOSE:  Perform administrative tasks such as update the
 *                                    registry, read performace counters,
control services,
 *                                    etc.
 *
 *            Author:                 Bill Carr
 *
              carr@perfom.enet.dec.com
 */


#define MAX_WEB_PAGE_SIZE 3096

#define GET_REGISTRY_KEY_LOCATOR
              "<tr><td>Key:</td><td>"
#define GET_REGISTRY_VALUE_LOCATOR
              "<tr><td>Value:</td><td>"
#define GET_REGISTRY_TYPE_LOCATOR
              "<tr><td>Type:</td><td>"
#define GET_REGISTRY_DATA_LOCATOR     "<tr><td
valign=top>Data:</td><td>"

#define GET_REGISTRY_TERMINATOR                         "</td></tr>"

#define GET_REGISTRY_KEY_TERMINATOR
              GET_REGISTRY_TERMINATOR
#define GET_REGISTRY_VALUE_TERMINATOR
              GET_REGISTRY_TERMINATOR
#define GET_REGISTRY_TYPE_TERMINATOR
              GET_REGISTRY_TERMINATOR
#define GET_REGISTRY_DATA_TERMINATOR
              GET_REGISTRY_TERMINATOR



 char                                 GetRegistrySuccessFmt[] = \
   "<table border>"
   "<tr>"
   "<th>Parameters</th>"
   "<tr>"
   "<td>"
     "<table>"
              GET_REGISTRY_KEY_LOCATOR"%s"GET_REGISTRY_K
EY_TERMINATOR
              GET_REGISTRY_VALUE_LOCATOR"%s"GET_REGISTR
Y_VALUE_TERMINATOR
              GET_REGISTRY_TYPE_LOCATOR"%s"GET_REGISTRY_
TYPE_TERMINATOR
              GET_REGISTRY_DATA_LOCATOR"%s"GET_REGISTRY_
DATA_TERMINATOR
     "</table>"
   "</td>"
   "</tr>"
   "</table>"
   ;

#define SERVICE_CONTROL_STATE_LOCATOR
              "<tr><td>State:</td><td>"
#define SERVICE_CONTROL_STATE_TERMINATOR "</td></tr>"

 char                                 ServiceControlSuccessFmt[] = \
   "<table border>"
   "<tr>"
   "<th>Parameters</th>"
   "<tr>"
   "<td>"
    "<table>"
              "<tr>"
              "<td>Computer:</td>"
              "<td>%s</td>"
              "</tr>"
              "<tr>"
              "<td>Service:</td>"
              "<td>%s</td>"
              "</tr>"
              "<tr>"
              "<td>Action:</td>"
              "<td>%s</td>"
              "</tr>"
              "<tr>"
              "<td>Executable:</td>"
              "<td>%s</td>"
              "</tr>"
              SERVICE_CONTROL_STATE_LOCATOR"%s"SERVICE_C
ONTROL_STATE_TERMINATOR
     "</table>"
   "</td>"
   "</tr>"
   "</table>"
   ;

#define END_BODY_STR "</BODY>"
#define REBOOT_SYNC_STR END_BODY_STR
#define SERVICE_QUERY_SYNC_STR END_BODY_STR
#define SERVICE_START_SYNC_STR END_BODY_STR
#define SERVICE_STOP_SYNC_STR END_BODY_STR
#define GET_REGISTRY_SYNC_STR END_BODY_STR
#define SET_REGISTRY_SYNC_STR END_BODY_STR
#define GET_PERF_CNTR_SYNC_STR END_BODY_STR

#define       REBOOT_STR \
"GET /admin.dll?CMD=ShutdownSystem&machine=&message=&"\
"delay="DELAY_BEFORE_REBOOT"&force=1&reboot=1 "\
"HTTP/1.0\nConnection: Keep-Alive\n\n"
#define SERVICE_QUERY_STR \
"GET
/admin.dll?CMD=ServiceControl&machine=&service=w3svc&action=quer
y "\
"HTTP/1.0\nConnection: Keep-Alive\n\n"
#define SERVICE_START_STR \
"GET
/admin.dll?CMD=ServiceControl&machine=&service=w3svc&action=start
"\
"HTTP/1.0\nConnection: Keep-Alive\n\n"
#define SERVICE_STOP_STR \
"GET
/admin.dll?CMD=ServiceControl&machine=&service=w3svc&action=stop
"\
"HTTP/1.0\nConnection: Keep-Alive\n\n"
#define GET_REGISTRY_STR \
"GET /admin.dll?CMD=GetRegistry&key=%s&value=%s "\
"HTTP/1.0\nConnection: Keep-Alive\n\n"
#define SET_REGISTRY_STR \
"GET
/admin.dll?CMD=SetRegistry&key=%s&value=%s&type=%s&data=%s "\
"HTTP/1.0\nConnection: Keep-Alive\n\n"
#define GET_PERF_CNTR_STR \
"GET %s?CMD=GetPerformanceCounter&object=HTTP Service&"\
"counter=Current Connections HTTP/1.0\nConnection: Keep-Alive\n\n"

#define SERVICE_CONTROL_STATE_STOPPED
              "Stopped"
#define SERVICE_CONTROL_STATE_START_PENDING   "Start
pending"
#define SERVICE_CONTROL_STATE_STOP_PENDING     "Stop
pending"
#define SERVICE_CONTROL_STATE_RUNNING
              "Running"
#define SERVICE_CONTROL_STATE_CONTINUE_PENDING
              "Continue pending"
#define SERVICE_CONTROL_STATE_PAUSE_PENDING   "Pause
pending"
#define SERVICE_CONTROL_STATE_PAUSED
              "Paused"
```

#endif

ckpt.c

```
/*+*************************************************************
**************
 *                                                             *
 *  COPYRIGHT (c) 1997, 2000 BY                                *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                               *
 *                                                             *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                             *
 *                                                             *

*************************************************************
*************-*/
/*+
 * Abstract: This file contains the Digital created front end functions
 *                              for the tpcc benchmark.
 *
 * Author: W Carr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
*/

#ifdef _WIN32
# include <windows.h>
# include <winsock2.h>
#endif
#include <stdio.h>

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

#include <ckpt.h>

static BOOL             gbCKPTInitialized = FALSE;
static DBContext        gDBC = INVALID_DB_CONTEXT;

#ifdef USE_PROCESSES
# define InitializeCriticalSection(Section)
# define EnterCriticalSection(Section)
# define LeaveCriticalSection(Section)
# define DeleteCriticalSection(Section)
#else
static CRITICAL_SECTION         CKPTCriticalSection;
#endif

int
TPCCCheckpointConnect( pLoginData *ppLogin )
{
  int                           retcode = ERR_DB_SUCCESS;

  if( ! gbCKPTInitialized ) {
    InitializeCriticalSection( &CKPTCriticalSection );

    retcode = (*ppLogin)->status = TPCCConnectDB( &gDBC, *ppLogin );

    TPCCLog( "%s, dbprocptr = %8X\r\n",
                    (*ppLogin)->databaseLogin.szApplication, gDBC );

    gbCKPTInitialized = TRUE;
  }

  return retcode;
}

int
TPCCCheckpoint( pCheckpointData *ppCheckpoint )
{
  int retcode;

  if( ! gbCKPTInitialized ) {
    retcode = ERR_CKPT_NOT_INITIALIZED;
  }
  else {
    EnterCriticalSection( &CKPTCriticalSection );
    retcode = TPCCCheckpointDB( gDBC, *ppCheckpoint );
    LeaveCriticalSection( &CKPTCriticalSection );
  }

  return retcode;
}

int
TPCCCheckpointDisconnect( pConnData *ppConn )
{
  gbCKPTInitialized = FALSE;

  DeleteCriticalSection( &CKPTCriticalSection );

  return TPCCDisconnectDB( gDBC, *ppConn );
}
```

ckpt.h

```
#ifndef CKPT_H
#define CKPT_H
/*+*************************************************************
**************
 *                                                             *
 *  COPYRIGHT (c) 1997, 2000 BY                                *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                               *
 *                                                             *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                             *
 *                                                             *
```

```
************************************************************
************-*/
/*+
 * Abstract: This file contains the Digital created front end functions
 *                              for the tpcc benchmark.
 *
 * Author: W Carr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
*/

#endif
```

config.c

```
/*+************************************************************
**************
 *                                                  *
 *  COPYRIGHT (c) 1997, 2000 BY                     *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.              *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                    *
 *                                                  *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.            *
 *                                                  *
 *                                                  *

************************************************************
************-*/

/*+
 * Abstract: This is the source file for globals.
 *
 * Author: WCarr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
-*/



#define CONFIG_C

#ifdef _WIN32
# include <windows.h>
# include <winsock2.h>
#else
# include <unistd.h>
#endif

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

#define CONFIG_C
#include <config.h>


char *
ConfigCodeToStr( int Code )
{
  int                                   Index;

  Index = FFE_C_MAX_PARAM - FFE_C_BASE;
  while( 0 < Index && Code != ConfigStrs[Index].Code )
  {
    Index--;
  }

  return( ConfigStrs[Index].Str );
}

int
GetConfigStr( int Code, int *Len, char *pString )
{
  int                                   Status;

  Status = GetConfigValue( ConfigCodeToStr( Code ), Len, pString );
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  return( ERR_SUCCESS );
}

int
GetConfigStrBool( int Code, BOOL *pBool )
{
  char                                  TempStr[FILENAMESIZE];
  int                                   Status;
  char                                  FirstLetter;
  int                                   Len = sizeof( TempStr ) - 1;

  Status = GetConfigValue( ConfigCodeToStr( Code ), &Len, TempStr );
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  FirstLetter = TempStr[0];

  switch (FirstLetter)
  {
    case 't':
    case 'T':
    case '1':
      *pBool = TRUE;
      break;

    case 'f':
    case 'F':
    case '0':
    default:
      *pBool = FALSE;
      break;
    }

  return( ERR_SUCCESS );
}

int
GetConfigStrDouble( int Code, double *pDouble )
{
  char                                  TempStr[FILENAMESIZE];
  int                                   Status;
  int                                   Len = sizeof( TempStr ) - 1;

  Status = GetConfigValue( ConfigCodeToStr( Code ), &Len, TempStr );
```

```
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  *pDouble = atof( TempStr );

  return( ERR_SUCCESS );
}

int
GetConfigStrInt( int Code, int *pInt )
{
  char                                  TempStr[FILENAMESIZE];
  int                                   Status;
  int                                   Len = sizeof( TempStr ) - 1;

  Status = GetConfigValue( ConfigCodeToStr( Code ), &Len, TempStr );
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  *pInt = atoi( TempStr );

  return( ERR_SUCCESS );
}

int
TPCCGetConfig( BOOL *pbLog,
                int *pPathLen, char *Path,
                pLoginData pLogin, int *pLoginDelay,
                pConfigData pConfig, pTransportData pTransport,
                pDeliveryTransportData pDeliveryTransport )
{
  int                                   Status;

  Status = GetConfigData( pConfig );
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetMiscData( pbLog, pPathLen, Path, pLoginDelay );
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetLoginData( pLogin );
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetTransportData( pTransport );
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetDeliveryTransportData( pDeliveryTransport );
  if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  return( ERR_SUCCESS );
}


int
GetConfigData( pConfigData pConfig )
{
  int                                   Status;

  Status = GetConfigStrInt( TOTAL_DISTRICT_COUNT, &pConfig-
>maxDistricts );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_TOTAL_DISTRICT_COUNT_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetConfigStrInt( MAXIMUM_CONNECTIONS, &pConfig-
>maxConnections );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_MAXIMUM_CONNECTIONS_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  return( ERR_SUCCESS );
}


int
GetDeliveryTransportData( pDeliveryTransportData pDeliveryTransport )
{
  int                                   Status;

  Status = GetConfigStrInt( NUMBER_OF_DELIVERY_THREADS,
                                        &pDeliveryTransport-
>num_threads );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return(
ERR_CANT_FIND_NUMBER_OF_DELIVERY_THREADS_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetConfigStrInt(
NUMBER_OF_QUEUED_DELIVERY_TRANSACTIONS,
                                        &pDeliveryTransport-
>num_queued_deliveries );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return(
ERR_CANT_FIND_NUMBER_OF_QUEUED_DELIVERY_TRANSACTI
ONS_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetConfigStrBool( USE_DELIVERY_PIPE,
                                        &pDeliveryTransport->use_pipe );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_USE_DELIVERY_PIPE_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetConfigStrBool( DISPLAY_DELIVERY_COMPLETIONS,
                                        &pDeliveryTransport-
>display_completions );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return(
ERR_CANT_FIND_DISPLAY_DELIVERY_COMPLETIONS_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }
```

```
  Status = GetConfigStrBool(
DELIVERY_SERVER_USES_TRANSACTION_SERVER,
                                        &pDeliveryTransport-
>use_transport );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return(
ERR_CANT_FIND_DELIVERY_SERVER_USES_TRANSACTION_SER
VER_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetConfigStrBool( FLUSH_DELIVERY_LOG,
                                        &pDeliveryTransport->flush_log
);
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_FLUSH_DELIVERY_LOG_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetConfigStrBool( WRITE_DELIVERY_LOG,
                                        &pDeliveryTransport->write_log
);
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_WRITE_DELIVERY_LOG_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  if( pDeliveryTransport->display_completions && !pDeliveryTransport-
>use_pipe )
    pDeliveryTransport->num_queued_responses =
      pDeliveryTransport->num_queued_deliveries;
  else
    pDeliveryTransport->num_queued_responses = 0;

  return( ERR_SUCCESS );
}


int
GetLoginData( pLoginData pLogin )
{
  int                                   Status;
  char                                  tmp[16];
  int                                   id;
  int                                   size;

  size = sizeof( pLogin->databaseLogin.szUser );
  Status = GetConfigStr( USER, &size, pLogin->databaseLogin.szUser );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_USER_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  size = sizeof( pLogin->databaseLogin.szPassword );
  Status = GetConfigStr( PASSWORD, &size, pLogin-
>databaseLogin.szPassword );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_PASSWORD_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  size = sizeof( pLogin->databaseLogin.szDatabase );
  Status = GetConfigStr( DATABASE, &size, pLogin-
>databaseLogin.szDatabase );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_DATABASE_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  size = sizeof( pLogin->databaseLogin.szServer );
  Status =  GetConfigStr( SERVER, &size, pLogin->databaseLogin.szServer
);
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_SERVER_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }


  gethostname(tmp, sizeof(tmp));
#ifdef _WIN32
  id = GetCurrentThreadId( );
#else
  id = getpid( );
#endif
  sprintf( pLogin->databaseLogin.szApplication, "%s:TPCC - %d", tmp, id );

  Status = GetConfigStrInt( RUN_ID, &pLogin->databaseLogin.iRunId );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return ERR_CANT_FIND_RUN_ID_VALUE;
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  return( ERR_SUCCESS );
}


int
GetMiscData( BOOL *pbLog, int *pPathLen, char *Path, int *pLoginDelay
)
{
  int                                   Status;


  GetConfigStrBool( LOG, pbLog );

  Status = GetConfigStr( PATH, pPathLen, Path );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return ERR_CANT_FIND_PATH_VALUE;
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetConfigStrInt( LOGIN_DELAY, pLoginDelay );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    *pLoginDelay = 0;
  }

  return( ERR_SUCCESS );
}

int
GetLoginDataCheckpoint( pLoginData pLogin )
{
  int                                   Status;
```

```
  char                                  tmp[16];
  int                                   id;
  int                                   size1;
  int                                   size2;

  size1 = size2 = sizeof( pLogin->databaseLogin.szUser );
  Status = GetConfigStr( CKPT_USER, &size1, pLogin-
>databaseLogin.szUser );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    Status = GetConfigStr( USER, &size1, pLogin->databaseLogin.szUser );
  }
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_USER_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  size1 = size2 = sizeof( pLogin->databaseLogin.szPassword );
  Status = GetConfigStr( CKPT_PASSWORD, &size1,
                                        pLogin->databaseLogin.szPassword
);
  if( ERR_CANT_FIND_VALUE == Status )
  {
    Status = GetConfigStr( PASSWORD, &size1, pLogin-
>databaseLogin.szPassword );
  }
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_PASSWORD_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  size1 = sizeof( pLogin->databaseLogin.szDatabase );
  Status = GetConfigStr( DATABASE, &size1, pLogin-
>databaseLogin.szDatabase );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_DATABASE_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  size1 = sizeof( pLogin->databaseLogin.szServer );
  Status =  GetConfigStr( SERVER, &size1, pLogin-
>databaseLogin.szServer );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return( ERR_CANT_FIND_SERVER_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }


  gethostname(tmp, sizeof(tmp));
#ifdef _WIN32
  id = GetCurrentThreadId( );
#else
  id = getpid( );
#endif
  sprintf( pLogin->databaseLogin.szApplication, "%s:CP - %d", tmp, id );

  Status = GetConfigStrInt( RUN_ID, &pLogin->databaseLogin.iRunId );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return ERR_CANT_FIND_RUN_ID_VALUE;
  }

  return( ERR_SUCCESS );
}

int
GetTransportData( pTransportData pTransport )
{
  int                                   Status;

  Status = GetConfigStrBool(
ASYNCHRONOUS_TRANSACTION_SERVERS,
                                        &pTransport->asynchronous );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return(
ERR_CANT_FIND_ASYNCHRONOUS_TRANSACTION_SERVERS_V
ALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  Status = GetConfigStrBool( GENERIC_TRANSACTION_SERVERS,
                                        &pTransport->generic );
  if( ERR_CANT_FIND_VALUE == Status )
  {
    return(
ERR_CANT_FIND_GENERIC_TRANSACTION_SERVERS_VALUE );
  }
  else if( ERR_SUCCESS != Status )
  {
    return( Status );
  }

  if( pTransport->generic )
  {
    Status = GetConfigStrInt( NUMBER_OF_GC_SERVERS, &pTransport-
>num_gc );
    if( ERR_CANT_FIND_VALUE == Status )
    {
      return( ERR_CANT_FIND_NUMBER_OF_GC_SERVERS_VALUE );
    }
    else if( ERR_SUCCESS != Status )
    {
      return( Status );
    }
    pTransport->num_dy = 0;
    pTransport->num_no = 0;
    pTransport->num_os = 0;
    pTransport->num_pt = 0;
    pTransport->num_sl = 0;
  }
  else
  {
    Status = GetConfigStrInt( NUMBER_OF_DY_SERVERS, &pTransport-
>num_dy );
    if( ERR_CANT_FIND_VALUE == Status )
    {
      return( ERR_CANT_FIND_NUMBER_OF_DY_SERVERS_VALUE );
    }
    else if( ERR_SUCCESS != Status )
    {
      return( Status );
    }
    Status = GetConfigStrInt( NUMBER_OF_NO_SERVERS, &pTransport-
>num_no );
    if( ERR_CANT_FIND_VALUE == Status )
    {
      return( ERR_CANT_FIND_NUMBER_OF_NO_SERVERS_VALUE );
    }
    else if( ERR_SUCCESS != Status )
    {
      return( Status );
    }
    Status = GetConfigStrInt( NUMBER_OF_OS_SERVERS, &pTransport-
>num_os );
    if( ERR_CANT_FIND_VALUE == Status )
    {
      return( ERR_CANT_FIND_NUMBER_OF_OS_SERVERS_VALUE );
    }
    else if( ERR_SUCCESS != Status )
```

```
    {
     return( Status );
    }
    Status = GetConfigStrInt( NUMBER_OF_PT_SERVERS, &pTransport-
>num_pt );
    if( ERR_CANT_FIND_VALUE == Status )
    {
     return( ERR_CANT_FIND_NUMBER_OF_PT_SERVERS_VALUE );
    }
    else if( ERR_SUCCESS != Status )
    {
     return( Status );
    }
    Status = GetConfigStrInt( NUMBER_OF_SL_SERVERS, &pTransport-
>num_sl );
    if( ERR_CANT_FIND_VALUE == Status )
    {
     return( ERR_CANT_FIND_NUMBER_OF_SL_SERVERS_VALUE );
    }
    else if( ERR_SUCCESS != Status )
    {
     return( Status );
    }
    pTransport->num_gc = 0;
  }

 return( ERR_SUCCESS );
}
```

config.h

```
#ifndef CONFIG_H
#define CONFIG_H

/*+************************************************************
**************
 *                                                            *
 *  COPYRIGHT (c) 1997, 2000 BY                               *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                              *
 *                                                            *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                            *
 *                                                            *

************************************************************
************-*/

/*+
 * Abstract: This is the header file for config.c.
 *
 * Author: WCarr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
 */

#define FFE_C_BASE
                300000

#define UNDEFINED_CONFIG_PARAM
                (FFE_C_BASE+0)
#define ASYNCHRONOUS_TRANSACTION_SERVERS
                (FFE_C_BASE+1)
#define CKPT_PASSWORD
                (FFE_C_BASE+2)
#define CKPT_USER
                (FFE_C_BASE+3)
#define DATABASE
                (FFE_C_BASE+4)
#define DELIVERY_SERVER_USES_TRANSACTION_SERVER
                (FFE_C_BASE+5)
#define DISPLAY_DELIVERY_COMPLETIONS
                (FFE_C_BASE+6)
#define FLUSH_DELIVERY_LOG
                (FFE_C_BASE+7)
#define GENERIC_TRANSACTION_SERVERS
                (FFE_C_BASE+8)
#define LOG
                (FFE_C_BASE+9)
#define LOGIN_DELAY
                (FFE_C_BASE+10)
#define MAXIMUM_CONNECTIONS
                (FFE_C_BASE+11)
#define TOTAL_DISTRICT_COUNT
                (FFE_C_BASE+12)
#define NUMBER_OF_DELIVERY_THREADS
                (FFE_C_BASE+13)
#define NUMBER_OF_DY_SERVERS
                (FFE_C_BASE+14)
#define NUMBER_OF_GC_SERVERS
                (FFE_C_BASE+15)
#define NUMBER_OF_NO_SERVERS
                (FFE_C_BASE+16)
#define NUMBER_OF_OS_SERVERS
                (FFE_C_BASE+17)
#define NUMBER_OF_PT_SERVERS
                (FFE_C_BASE+18)
#define NUMBER_OF_QUEUED_DELIVERY_TRANSACTIONS
                (FFE_C_BASE+19)
#define NUMBER_OF_SL_SERVERS
                (FFE_C_BASE+20)
#define PASSWORD
                (FFE_C_BASE+21)
#define PATH
                (FFE_C_BASE+22)
#define RUN_ID
                (FFE_C_BASE+23)
#define SERVER
                (FFE_C_BASE+24)
#define FFE_TM_MODULE
                (FFE_C_BASE+25)
#define USER
                (FFE_C_BASE+26)
#define USE_DELIVERY_PIPE
                (FFE_C_BASE+27)
#define WRITE_DELIVERY_LOG
                (FFE_C_BASE+28)

#define FFE_C_MAX_PARAM
                (FFE_C_BASE+28)


typedef struct _ffe_config_str_t
{
  int                   Code;
  char                  Str[256];
} ffe_config_str_t;

#ifdef CONFIG_C

ffe_config_str_t ConfigStrs[] =
{
  { UNDEFINED_CONFIG_PARAM,             "Unknown config
parameter" },
```

```
  { ASYNCHRONOUS_TRANSACTION_SERVERS,
"AsynchronousTransactionServers" },
  { CKPT_PASSWORD,                  "CkptPassword" },
  { CKPT_USER,                      "CkptUser" },
  { DATABASE,                       "Database" },
  {
DELIVERY_SERVER_USES_TRANSACTION_SERVER,"DeliveryServe
rUsesTransactionServer" },
  { DISPLAY_DELIVERY_COMPLETIONS,
"DisplayDeliveryCompletions" },
  { FLUSH_DELIVERY_LOG,             "FlushDeliveryLog" },
  { GENERIC_TRANSACTION_SERVERS,
"GenericTransactionServers" },
  { LOG,                            "Log" },
  { LOGIN_DELAY,                    "DBLoginDelay" },
  { MAXIMUM_CONNECTIONS,            "MaxConnections" },
  { TOTAL_DISTRICT_COUNT,
"TOTAL_DISTRICT_COUNT" },
  { NUMBER_OF_DELIVERY_THREADS,
"NumberOfDeliveryThreads" },
  { NUMBER_OF_DY_SERVERS,           "NumberOfDYServers" },
  { NUMBER_OF_GC_SERVERS,           "NumberOfGCServers" },
  { NUMBER_OF_NO_SERVERS,           "NumberOfNOServers" },
  { NUMBER_OF_OS_SERVERS,           "NumberOfOSServers" },
  { NUMBER_OF_PT_SERVERS,           "NumberOfPTServers" },
  {
NUMBER_OF_QUEUED_DELIVERY_TRANSACTIONS,"NumberOfQu
euedDeliveryTransactions" },
  { NUMBER_OF_SL_SERVERS,           "NumberOfSLServers" },
  { PASSWORD,                       "Password" },
  { PATH,                           "Path" },
  { RUN_ID,
"RUN_ID" },
  { SERVER,                         "Server" },
  { FFE_TM_MODULE,                  "TM" },
  { USER,                           "User" },
  { USE_DELIVERY_PIPE,              "UseDeliveryPipe" },
  { WRITE_DELIVERY_LOG,             "WriteDeliveryLog" }
};
#else
extern ffe_config_str_t ConfigStrs[];
#endif



char *ConfigCodeToStr( int Code );
int GetConfigStr( int Code, int *Len, char *pString );
int GetConfigStrBool( int Code, BOOL *pBool );
int GetConfigStrDouble( int Code, double *pDouble );
int GetConfigStrInt( int Code, int *pInt );
int GetConfigValue( char *Name, int *Len, char *Value );
int TPCCGetConfig( BOOL *pbLog,
                   int *pPathLen, char *Path,
                   pLoginData pLogin, int *pLoginDelay,
                   pConfigData pConfig, pTransportData
pTransport,
                   pDeliveryTransportData pDeliveryTransport );
int GetConfigData( pConfigData pConfig );
int GetLoginData( pLoginData pLogin );
int GetLoginDataCheckpoint( pLoginData pLogin );
int GetMiscData( BOOL *pbLog, int *pPathLen, char *Path, int
*pLoginDelay );
int GetDeliveryTransportData( pDeliveryTransportData pDeliveryTransport
);
int GetTransportData( pTransportData pTransport );
int SetConfigValue( char *Name, char *Value );

#endif
```

configdb.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
************-*/

/*+
 * Abstract: This is the source file for configuration database manipulation.
 *
 * Author: WCarr
 * Creation Date: October 1999
 *
 * Modified history:
 *
 *
-*/

#ifdef _WIN32
# include <windows.h>
#endif

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

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

#define FFE_HOME "FFE"

#define CONFIG_STR_SIZE 4096

typedef struct _config_entry_t *config_entry_ptr_t;
typedef struct _config_entry_t
{
 config_entry_ptr_t        Next;
 char                      Name[CONFIG_STR_SIZE];
 char                      Value[CONFIG_STR_SIZE];
} config_entry_t;


void ParseEntry( char *Entry, char **EntryName, int *Len, char
**EntryValue );
int  OpenConfigFile( FILE **FilePtrPtr, char *Mode );


int
GetConfigValue( char *Name, int *Len, char *Value )
{
 char                      Entry[CONFIG_STR_SIZE];
 int                       EntryLen;
 char                      *EntryName;
 char                      *EntryValue;
 FILE                      *FilePtr;
 int                       Status;
```

```
  Status = ERR_SUCCESS;

  Status = OpenConfigFile( &FilePtr, "r" );

  if( ERR_SUCCESS == Status )
  {

   Status = ERR_CANT_FIND_VALUE;
   while( !feof( FilePtr ) && ERR_SUCCESS != Status )
   {
    fgets( Entry, sizeof( Entry ), FilePtr );

    if( !feof( FilePtr ))
    {
                ParseEntry( Entry, &EntryName, &EntryLen, &EntryValue );


                if( 0 == strcmp( EntryName, Name ))
                {
                 Status = ERR_SUCCESS;
                 *Len = EntryLen;
                 strcpy( Value, EntryValue );
                }
    }
   }

   fclose( FilePtr );
  }
  else
  {
   *Value = '\0';
   *Len = 0;
  }

  return( Status );
}

int
OpenConfigFile( FILE **FilePtrPtr, char *Mode )
{
  char                          *Env;
  char                          Path[PATH_MAX];
  int                           Status;

  Status = ERR_SUCCESS;


  Env = getenv( FFE_HOME );
  if( NULL == Env )
  {
   Status = ERR_HOME_ENV_VAL_NOT_SET;
  }

  if( ERR_SUCCESS == Status )
  {
   strcpy( Path, Env );
   strcat( Path, "/ffe.cfg" );

   *FilePtrPtr = fopen( Path, Mode );
   if( NULL == *FilePtrPtr )
   {
    Status = ERR_CANT_OPEN_CONFIG_DATABASE;
   }
  }

  return( Status );
}


int
SetConfigValue( char *Name, char *Value )
{
  config_entry_ptr_t    ConfigEntryPtr;
  config_entry_ptr_t    ConfigEntryFirstPtr;
  config_entry_ptr_t    ConfigEntryLastPtr;
  char                          Entry[CONFIG_STR_SIZE];
  int                           EntryLen;
  char                          *EntryName;
  char                          *EntryValue;
  FILE                          *FilePtr;
  int                           Status;

  Status = ERR_SUCCESS;

  Status = OpenConfigFile( &FilePtr, "r" );

  ConfigEntryFirstPtr = NULL;
  if( ERR_SUCCESS == Status )
  {

   Status = ERR_CANT_FIND_VALUE;
   while( !feof( FilePtr ))
   {
    fgets( Entry, sizeof( Entry ), FilePtr );

    if( !feof( FilePtr ))
    {
                ParseEntry( Entry, &EntryName, &EntryLen, &EntryValue );

                ConfigEntryPtr = malloc( sizeof( *ConfigEntryPtr ));
                ConfigEntryPtr->Next = NULL;

                if( NULL == ConfigEntryFirstPtr )
                {
                 ConfigEntryFirstPtr = ConfigEntryPtr;
                 ConfigEntryLastPtr = ConfigEntryPtr;
                }
                else
                {
                 ConfigEntryLastPtr->Next = ConfigEntryPtr;
                 ConfigEntryLastPtr = ConfigEntryPtr;
                }

                strcpy( ConfigEntryPtr->Name, EntryName );


                if( 0 == strcmp( EntryName, Name ))
                {
                 Status = ERR_SUCCESS;
                 strcpy( ConfigEntryPtr->Value, Value );
                }
                else
                {
                 strcpy( ConfigEntryPtr->Value, EntryValue );
                }
    }
   }

   fclose( FilePtr );
  }

  if( ERR_CANT_FIND_VALUE == Status ||
ERR_CANT_OPEN_CONFIG_DATABASE == Status )
  {

   ConfigEntryPtr = malloc( sizeof( *ConfigEntryPtr ));
   ConfigEntryPtr->Next = NULL;

   if( NULL == ConfigEntryFirstPtr )
   {
    ConfigEntryFirstPtr = ConfigEntryPtr;
    ConfigEntryLastPtr = ConfigEntryPtr;
   }
   else
   {
    ConfigEntryLastPtr->Next = ConfigEntryPtr;
    ConfigEntryLastPtr = ConfigEntryPtr;
   }

   strcpy( ConfigEntryPtr->Name, Name );

   Status = ERR_SUCCESS;
   strcpy( ConfigEntryPtr->Value, Value );
  }


  Status = OpenConfigFile( &FilePtr, "w" );

  if( ERR_SUCCESS == Status )
  {
```

```
   do
   {
    ConfigEntryPtr = ConfigEntryFirstPtr;
    fprintf( FilePtr, "%s %s\n",
                    ConfigEntryPtr->Name, ConfigEntryPtr->Value );

    ConfigEntryFirstPtr = ConfigEntryPtr->Next;
    free( ConfigEntryPtr );
   } while( NULL != ConfigEntryFirstPtr );

   fclose( FilePtr );
  }

  return( Status );
}


void
ParseEntry( char *Entry, char **EntryName, int *Len, char **EntryValue )
{
  char                                  *ChrPtr;

  ChrPtr = Entry;


  while( ' ' == *ChrPtr || '\t' == *ChrPtr || '\n' == *ChrPtr )
  {
   ChrPtr++;
  }

  *EntryName = ChrPtr;


  while( ' ' != *ChrPtr && '\t' != *ChrPtr && '\n' != *ChrPtr )
  {
   ChrPtr++;
  }


  if( '\n' == *ChrPtr )
  {
   *ChrPtr++ = '\0';
   *ChrPtr = '\n';
  }
  else
  {
   *ChrPtr++ = '\0';
  }


  while( ' ' == *ChrPtr || '\t' == *ChrPtr )
  {
   ChrPtr++;
  }

  *EntryValue = ChrPtr;


  while( ' ' != *ChrPtr && '\t' != *ChrPtr && '\n' != *ChrPtr )
  {
   ChrPtr++;
  }

  *ChrPtr = '\0';
  *Len = ChrPtr - *EntryValue;
}
```

confvar.h

```
#ifndef CONFVAR_H
#define CONFVAR_H


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

struct _DeliveryTransportData {
  BOOL                  display_completions;
  BOOL                  flush_log;
  int                   num_threads;
  int                   num_queued_deliveries;
  int                   num_queued_responses;
  BOOL                  use_transport;
  BOOL                  use_pipe;
  BOOL                  write_log;
};
typedef struct _DeliveryTransportData  DeliveryTransportData,
*pDeliveryTransportData;


struct _DatabaseLoginData {
  char                  szServer[32];
  char                  szDatabase[32];
  char                  szUser[32];
  char                  szPassword[32];
  char                  szApplication[32];
  int                   iRunId;
  int                   iLoginDelay;
};
typedef struct _DatabaseLoginData DatabaseLoginData,
*pDatabaseLoginData;


struct _ConfigData {
  int                                   maxConnections;
  int                                   maxDistricts;
};
typedef struct _ConfigData ConfigData, *pConfigData;

#endif
```

crestdl.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                  *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                          *
 *                                                          *

**********************************************************
*************-*/
#ifdef _WIN32
```

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

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


#include <tpccstruct.h>
#include <tpcc_acmsxp.h>

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

void usage( char *prog );

int
main(int argc, char *argv[])
{
 FILE *fptr;
 char *dir;
 int  iMaxSize;

 dir = getenv( "USR_OBJ" );

 if( 2 != argc ) {
  usage( argv[0] );
 }

 if( NULL == ( fptr = fopen( argv[1],"w" )))
 {
  fprintf( stderr, "\nCould not open file %s\n", argv[1] );
  exit( 1 );
 }

 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", argv[1] );

 return 0;
}

void
usage( char *prog )
{
 fprintf( stderr, "usage: %s <output file name>\n", prog );

 exit( 1 );
}
```

deli_cli.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
*************-*/

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

#define DELI_CLI_C

#ifdef _WIN32
# include <windows.h>
#else
# include <unistd.h>
# include <errno.h>
# include <sys/time.h>
#endif

#include <time.h>

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

#ifdef _WIN32
# include <buf.h>
#endif
#include <deli_srv.h>
#include <transpool.h>

#ifdef _WIN32
static BOOL gbUsePipe = FALSE;
#else
static BOOL gbUsePipe = TRUE;
#endif

static BOOL gbDisplayCompletions = TRUE;
```

```
int
DELIClientStartup( pDeliveryTransportData pDeliveryTransport,
                                pLoginData pLogin, int loginDelay, char *Path )
{
 int status;

#ifdef _WIN32

 gbUsePipe = pDeliveryTransport->use_pipe;
 if( gbUsePipe )
 {
  gbDisplayCompletions = FALSE;
 }
 else
 {
  gbDisplayCompletions = pDeliveryTransport->display_completions;
 }
#endif

 status = DELIServerStartup( pDeliveryTransport, pLogin, loginDelay, Path
);
 if( ERR_SUCCESS != status )
  return( status );

 return( ERR_SUCCESS );
}



int
DELIClientShutdown( void )
{
 return( DELIServerShutdown( ));
}


int
TPCCDelivery( pDeliveryData *ppDelivery,
                pDeliveryData
pCompletedDeliveries[DELIVERY_RESPONSE_COUNT] )
{
 pDeliveryData                          pDeliveryCopy;
 size_t                         bw;
 size_t                         br;
 int                                    ii;
 int                                    status;



 ii = 0;
 if( gbDisplayCompletions ) {
  while( ii < DELIVERY_RESPONSE_COUNT ) {
#ifdef _WIN32
    status = bufread( &pCompletedDeliveries[ii], sizeof( pDeliveryData ),
                                        &br, 0, outputbuf );
    if( BUF_READTIMEOUT == status ) {

            break;
    }
    else if( BUF_SUCCESS != status )
            return ERR_DELIVERY_OUTPUT_PIPE_READ;
#else
    pCompletedDeliveries[ii] = ReserveTransactionStruct(
DELIVERY_TRANS );
    br = sizeof( DeliveryData );
    status = DELIQueueRead( outputbuf,
                                        (void *)pCompletedDeliveries[ii],
&br, FALSE );
    if( ERR_SUCCESS != status )
    {
            return( status );
    }
    if( 0 == br )
    {
            UnreserveTransactionStruct( DELIVERY_TRANS,
&pCompletedDeliveries[ii] );
            break;
    }
#endif
    ii++;
  }
 }


 while( ii < DELIVERY_RESPONSE_COUNT )
 {
  pCompletedDeliveries[ii++] = NULL;
 }


 if( 0 == (*ppDelivery)->queue_time )
  time( &(*ppDelivery)->queue_time );



 StartTime( (*ppDelivery)->delta_time );

 if( gbUsePipe ) {
#ifdef _WIN32
   if( 0 == WriteFile( ghPipeInputWrite, (*ppDelivery),
                                        sizeof(DeliveryDataInput), &bw,
NULL )) {
    status = GetLastError( );
    return ERR_DB_DELIVERY_NOT_QUEUED;
   }
#else
#pragma message ("FIXME: When the delivery queue fills on Unix, you can
reach a deadlock.  There are two ways around this.  First, don't block and
send an error message to the caller that they should set the msg-tql parameter
in the ipc section of the sysconfigtab.  Second, use mmap() to create a shared
memory section and port the NT buf code so that the buffer size can be
managed by application code.")
   status = DELIQueueWrite( inputbuf, (void *)*ppDelivery,
                                        sizeof( DeliveryDataInput ), TRUE
);
   if( ERR_SUCCESS != status )
   {
    if( ERR_DB_DELIVERY_NOT_QUEUED == status )
    {
            TPCCErr( "Check the setting of msg-tql in the ipc: section of
the sysconfigtab.\r\n" );
    }
    return( status );
   }
#endif
  }
#ifdef _WIN32
 else {


   pDeliveryCopy = ReserveTransactionStruct( DELIVERY_TRANS );
   memcpy( (char *)pDeliveryCopy, (char *)(*ppDelivery),
            sizeof( DeliveryDataInput ));


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

 }
#endif



 TRANSACTION_DEBUG_STAGE( *ppDelivery,

IN_LH|IN_RH|IN_DB|LEAVING_DB|LEAVING_RH|LEAVING_LH );

 return ERR_DB_SUCCESS;
}
```

deli_cli.h

```
#ifndef DELI_CLI_H
```

```
#define DELI_CLI_H
/*+**********************************************************
**************
 *                                                          *
 * COPYRIGHT (c) 1997, 2000 BY                              *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 * 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
DIGITAL EQUIPMENT   *
 * CORPORATION.                                             *
 *                                                          *
 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS   *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                          *
 *                                                          *

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

int DELIClientStartup( pDeliveryTransportData pDeliveryTransport,
                                pLoginData pLogin, int loginDelay, char
*Path );
int DELIClientShutdown( void );

#endif
```

deli_srv.c

```
/*+**********************************************************
**************
 *                                                          *
 * COPYRIGHT (c) 1997, 2000 BY                              *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 * 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
DIGITAL EQUIPMENT   *
 * CORPORATION.                                             *
 *                                                          *
 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS   *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                          *
 *                                                          *

**********************************************************
*************-*/

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

#define DELI_SRV_C

#ifdef _WIN32
# include <windows.h>
# include <winsock.h>
# include <process.h>
#else
# include <unistd.h>
# include <errno.h>
# include <signal.h>
# include <fcntl.h>
# include <sys/file.h>
# include <sys/time.h>
# include <sys/wait.h>
# include <sys/msg.h>
# define __stdcall
# define GetCurrentThreadId getpid
#endif
#include <time.h>
#include <stdio.h>
#include <stdlib.h>
#ifdef _WIN32
# include <crtdbg.h>
# include <buf.h>
#endif

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

#include <transpool.h>
#include <deli_srv.h>

static int      giDeliInitStatus = ERR_SUCCESS;
static long     gDeliThreadStartCtr;
#ifdef USE_PROCESSES
static BOOL gbUsePipe = TRUE;
#else
static BOOL gbUsePipe = FALSE;
static HANDLE           gDeliThreadStartEvent;

static HANDLE           hPipeInputRead = INVALID_HANDLE_VALUE;
static BOOL bDone;
#endif
static BOOL gbDisplayCompletions = TRUE;

static FILE   *fpLog = NULL;

static BOOL gbWriteDeliLog = FALSE;
static BOOL gbFlushDeliLog = FALSE;
static BOOL gbDirectConnect = FALSE;


void                    DELIErrorMessage(int iError);
void                    DELILog( pDeliveryData pDelivery );
```

```
unsigned      __stdcall DELIThread( void *ptr );
#ifdef USE_PROCESSES
# define      DELIVERY_MESSAGE 1
# define      DELIVERY_SHUTDOWN -1
# define      DELIVERY_STARTUP_SUCCESS -2
# define      DELIVERY_STARTUP_FAILURE -3
# define      INPUT_QUEUE_ID 71759
# define      OUTPUT_QUEUE_ID 10492
# define      DELI_PARAMS_NAME "/tmp/deli_server.params"
# define      DELI_LOCK_NAME "/tmp/deli_server.lock"
typedef struct _DELIMsgStruct
{
  long int                    iType;
  DeliveryData                Delivery;
} DELIMsgStruct, *pDELIMsgStruct;
typedef struct _DELIServerStruct
{
  int                         RunId;
  int                         CurrentServerCount;
  int                         CurrentClientCount;
} DELIServerStruct, *pDELIServerStruct;
int                           DELIServerShutdownInternal( pDELIServerStruct
pdss );
void                          DELISigchldHandler( int );
#if 1
# define      CRASH *segv = 0
int                           *segv = NULL;
#else
int                           num = 1;
int                           denom = 0;
# define      CRASH num/denom;
#endif
#endif



void
DELIErrorMessage(int iError)
{
  TPCCErr( "*Error(%d): %s\r\n", iError, TPCCErrString( iError ));
  return;
}



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);

  memcpy( &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 ( gbFlushDeliLog )
    fflush(fpLog);

  return;
}


#ifdef USE_PROCESSES
int
DELIQueueRead( int pipe, void *data, unsigned long *plen, BOOL block )
{
  int                         status;
  int                         err;
  DELIMsgStruct               DeliMsg;
  int                         retlen;
  int                         flags;

  flags = ( block ) ? 0 : IPC_NOWAIT;
  retlen =  msgrcv( pipe, &DeliMsg, *plen, DELIVERY_MESSAGE, flags );
  if( -1 == retlen && ENOMSG == errno )
  {
    *plen = 0;
    status = ERR_SUCCESS;
  }
  else if( -1 == retlen || *plen != retlen )
  {
    err = errno;
    status = ERR_DELIVERY_PIPE_READ;
  }
  else
  {
    memcpy( data, &DeliMsg.Delivery, retlen );
    status = ERR_SUCCESS;
  }

  return( status );
}
#endif


#ifdef USE_PROCESSES
int
DELIQueueWrite( int pipe, void *data, unsigned long len, BOOL block )
{
  int                         status;
  int                         err;
  DELIMsgStruct               DeliMsg;
  int                         retlen;
  int                         flags;

  DeliMsg.iType = DELIVERY_MESSAGE;
  memcpy( &DeliMsg.Delivery, data, len );

  flags = ( block ) ? 0 : IPC_NOWAIT;
  retlen = msgsnd( pipe, &DeliMsg, len, flags );
  if( -1 == retlen && EAGAIN == errno )
  {
    status = ERR_DB_DELIVERY_NOT_QUEUED;
  }
  else if( -1 == retlen )
  {
    err = errno;
CRASH;
    status = ERR_DB_DELIVERY_NOT_QUEUED;
  }
  else
  {
    status = ERR_SUCCESS;
  }

  return( status );
}
#endif


int
DELIServerShutdown( void )
{
```

```
  int                                   status;
#ifdef USE_PROCESSES
  int                                   err;
  int                                   lock;
  int                                   paramsFD;
  DELIServerStruct          dss;
  int                                   size;
#endif

  status = ERR_SUCCESS;

#ifdef USE_PROCESSES
  if( -1 == ( lock = open( DELI_LOCK_NAME, O_WRONLY | O_CREAT,
0666 )))
  {
    err = errno;
    status = ERR_DELIVERY_PARAMS_FILE;
  }
  else
  {
    flock( lock, LOCK_EX );

    if( -1 == ( paramsFD = open( DELI_PARAMS_NAME, O_RDWR, 0666
)))
    {
      status = ERR_DELIVERY_SHUTDOWN;
    }
    else
    {

      size = sizeof( DELIServerStruct );
      if( size != read( paramsFD, &dss, size ))
      {
                err = errno;
                status = ERR_DELIVERY_PARAMS_FILE;
      }
      else if( 0 == --dss.CurrentClientCount )
      {
                status = DELIServerShutdownInternal( &dss );
                unlink( DELI_PARAMS_NAME );
                unlink( DELI_LOCK_NAME );
      }
      else
      {
                if( (off_t) -1 == lseek( paramsFD, 0, SEEK_SET ) ||
                    size != write( paramsFD, &dss, size ))
                {
                  err = errno;
                  status = ERR_DELIVERY_PARAMS_FILE;
                }
      }
      close( paramsFD );
    }

    close( lock );
  }
#else
  if( gbUsePipe ) {
    CloseHandle( hPipeInputRead );
    CloseHandle( ghPipeInputWrite );
  }
  else {
    bufclose( inputbuf );
    bufclose( outputbuf );
  }
#endif

  if ( fpLog )
    fclose(fpLog);

  fpLog = NULL;

  return( status );
}

#ifdef USE_PROCESSES
int
DELIServerShutdownInternal( pDELIServerStruct pdss )
{
  int                                   status;
  int                                   inerr;
  int                                   outerr;
  int                                   ii;
  int                                   Count;
  DeliveryDataInput         DeliveryClose;

  if( -1 != inputbuf )
  {
    DeliveryClose.o_carrier_id = DELIVERY_SHUTDOWN;
    Count = pdss->CurrentServerCount;
    for( ii = 0; ii < Count; ii++ ) {
      status = DELIQueueWrite( inputbuf, (void *)&DeliveryClose,
                                                sizeof( DeliveryClose ), FALSE
);
      if( ERR_SUCCESS != status &&
ERR_DB_DELIVERY_NOT_QUEUED != status )
      {
                return( status );
      }
    }
    if( -1 == msgctl( inputbuf, IPC_RMID, NULL ))
    {
      inerr = errno;
CRASH;
      return( ERR_DELIVERY_PIPE_DESTROY );
    }
  }

#if 1
  if( -1 != outputbuf )
  {
    if( -1 == msgctl( outputbuf, IPC_RMID, NULL ))
    {
      outerr = errno;
CRASH;
      return( ERR_DELIVERY_PIPE_DESTROY );
    }
  }
#endif

  return( ERR_SUCCESS );
}
#endif



int
DELIServerStartup( pDeliveryTransportData pDeliveryTransport,
                                pLoginData pLogin, int loginDelay, char *Path )
{
  char                                  LogName[PATH_MAX];
#ifdef USE_PROCESSES
  int                                   ii;
  int                                   err;
  int                                   inerr;
  int                                   outerr;
  int                                   childPid;
  int                                   status;
  int                                   lock;
  int                                   paramsFD;
  DELIServerStruct          dss;
  int                                   ParamsSize = sizeof(
DELIServerStruct );
  int                                   size;
#else
  int                                   ii;
  int                                   iError;
  size_t                    inputbufsize;
  size_t                    outputbufsize;
  unsigned                  tid;
  unsigned long                         ulhThread;
  HANDLE                    hThread;
#endif

  gDeliThreadStartCtr = pDeliveryTransport->num_threads;
  gbWriteDeliLog = pDeliveryTransport->write_log;
  gbFlushDeliLog = pDeliveryTransport->flush_log;
  gbDirectConnect = !pDeliveryTransport->use_transport;
  strcat( strcpy( LogName, Path ), "delilog." );
```

```
#ifdef USE_PROCESSES
  status = ERR_SUCCESS;
  if( -1 == ( lock = open( DELI_LOCK_NAME, O_WRONLY | O_CREAT,
0666 )))
  {
    err = errno;
CRASH;
    status = ERR_DELIVERY_PARAMS_FILE;
  }
  else
  {
    flock( lock, LOCK_EX );

    if( -1 == ( paramsFD = open( DELI_PARAMS_NAME, O_RDWR |
O_CREAT, 0666 )))
    {
      err = errno;
      status = ERR_DELIVERY_PARAMS_FILE;
    }
    else
    {

      size = read( paramsFD, &dss, ParamsSize );
      if(( 0 != size ) && ( ParamsSize != size ))
      {

                status = ERR_DELIVERY_PARAMS_FILE;
      }
      else if(( 0 != size ) && ( pLogin->databaseLogin.iRunId == dss.RunId ))
      {

                dss.CurrentClientCount++;
      }
      else
      {
                status = ERR_SUCCESS;
                if(( 0 != size ) && ( pLogin->databaseLogin.iRunId !=
dss.RunId ))
                {

                  if( -1 == ( inputbuf = msgget( INPUT_QUEUE_ID, IPC_R )))
                  {
                    inerr = errno;
                  }
                  if( -1 == ( outputbuf = msgget( OUTPUT_QUEUE_ID,
IPC_R )))
                  {
                    outerr = errno;
                  }
                  if(( -1 == inputbuf && ENOENT != inerr ) ||
                     ( -1 == outputbuf && ENOENT != outerr ))
                  {
CRASH;
                    status = ERR_DELIVERY_PIPE_OPEN;
                  }
                  else if( -1 != inputbuf || -1 != outputbuf )
                  {
                    status = DELIServerShutdownInternal( &dss );
                  }
                  else
                  {
                    status = ERR_SUCCESS;
                  }
                }
                if( ERR_SUCCESS == status )
                {

                  if( -1 == msgget( INPUT_QUEUE_ID,
IPC_CREAT|IPC_EXCL|IPC_R|IPC_W ) ||
                      -1 == msgget( OUTPUT_QUEUE_ID,
IPC_CREAT|IPC_EXCL|IPC_R|IPC_W ))
                  {
                    err = errno;
CRASH;
                    status = ERR_DELIVERY_PIPE_CREATE;
                  }
                  else
                  {
                    signal( SIGCHLD, DELISigchldHandler );
                    for( ii = 0; ii < pDeliveryTransport->num_threads; ii++ )
                    {
                      if( -1 == ( childPid = fork( )))
                      {
                                err = errno;
                                status =
ERR_CANT_START_DELIVERY_THREAD;
                      }
                      else if( 0 < childPid )
                      {

                                Sleep( loginDelay );
                      }
                      else
                      {

                                close( paramsFD );
                                close( lock );
                                if( -1 == ( inputbuf = msgget(
INPUT_QUEUE_ID, IPC_R )) ||
                                    -1 == ( outputbuf = msgget(
OUTPUT_QUEUE_ID, IPC_W )))
                                {
CRASH;
                                  status = ERR_DELIVERY_PIPE_OPEN;
                                }

                                else if( gbWriteDeliLog && NULL ==
(fpLog=fopen(LogName, "wb")))
                                {
                                  status =
ERR_CANNOT_CREATE_RESULTS_FILE;
                                }
                                else
                                {
                                  status = DELIThread( pLogin );
                                }
                                exit( status );
                      }
                    }
                    dss.RunId = pLogin->databaseLogin.iRunId;
                    dss.CurrentServerCount = pDeliveryTransport-
>num_threads;
                    dss.CurrentClientCount = 1;
                  }
                }
      }

      if( (off_t) -1 == lseek( paramsFD, 0, SEEK_SET ) ||
                ParamsSize != write( paramsFD, &dss, ParamsSize ))
      {
                err = errno;
                status = ERR_DELIVERY_PARAMS_FILE;
      }
      close( paramsFD );
    }
    close( lock );
  }

  if( -1 == ( inputbuf = msgget( INPUT_QUEUE_ID, IPC_W )) ||
      -1 == ( outputbuf = msgget( OUTPUT_QUEUE_ID, IPC_R )))
  {
CRASH;
    status = ERR_DELIVERY_PIPE_OPEN;
  }

  return( status );
#else

  gbUsePipe = pDeliveryTransport->use_pipe;
  if( gbUsePipe )
  {
    gbDisplayCompletions = FALSE;
  }
  else
  {
    gbDisplayCompletions = pDeliveryTransport->display_completions;
  }


  if( gbWriteDeliLog ) {
```

```
    fpLog = fopen(LogName, "wb");
    if ( NULL == fpLog )
      return ERR_CANNOT_CREATE_RESULTS_FILE;
  }

  if( gbUsePipe ) {

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

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

    if( gbDisplayCompletions ) {

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


  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 < pDeliveryTransport->num_threads; ii++ ) {
      ulhThread = _beginthreadex( NULL, 0, DELIThread, pLogin, 0, &tid );
      if( 0 == ulhThread )
                return ERR_CANT_START_DELIVERY_THREAD;
      hThread = ( HANDLE )ulhThread;
      CloseHandle( hThread );
      if( !pDeliveryTransport->use_transport )
                Sleep( loginDelay );
    }
  }

  WaitForSingleObject( gDeliThreadStartEvent, INFINITE );

  return giDeliInitStatus;
#endif
}

#ifdef USE_PROCESSES
void
DELISigchldHandler( int sig )
{
  int           wait_status;


  while( wait3(&wait_status, WNOHANG, 0) > 0 ) {
    gDeliThreadStartCtr--;
  }
}
#endif



unsigned __stdcall
DELIThread( void *ptr )
{
  size_t                br;
  pDeliveryData                 pDelivery;
  int                           retcode;
  CallersContext        CC;
  pLoginData                    pCallersLogin;
  LoginData             login;
  char                          tmp[16];
  int                           status;
  DBContext             DBC;
#ifdef USE_PROCESSES
  BOOL                          bDone = FALSE;
#endif


  pCallersLogin = ptr;
  CC = 0;
  DBC = INVALID_DB_CONTEXT;



  if( gbDirectConnect ) {
    gethostname(tmp, sizeof(tmp));
    login.w_id = 0;
    login.ld_id = 0;
    login.CC = CC;
    strcpy( login.databaseLogin.szServer,
                pCallersLogin->databaseLogin.szServer );
    strcpy( login.databaseLogin.szDatabase,
                pCallersLogin->databaseLogin.szDatabase );
    strcpy( login.databaseLogin.szUser, pCallersLogin-
>databaseLogin.szUser );
    strcpy( login.databaseLogin.szPassword,
                pCallersLogin->databaseLogin.szPassword );
    sprintf( login.databaseLogin.szApplication, "%s:delisrv - %d",
                tmp, GetCurrentThreadId());
    status = TPCCConnectDB( &DBC, &login );
    if( ERR_DB_SUCCESS != status && ERR_SUCCESS !=
giDeliInitStatus )
    {
      giDeliInitStatus = status;
    }
  }

#ifdef USE_PROCESSES
#else

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

  WaitForSingleObject( gDeliThreadStartEvent, INFINITE );
#endif

  if( ERR_SUCCESS != giDeliInitStatus )
  {
    return( giDeliInitStatus );
  }


  while( !bDone ){

    if( gbUsePipe ) {
      pDelivery = ReserveTransactionStruct( DELIVERY_TRANS );
#ifdef USE_PROCESSES
      br = sizeof( DeliveryDataInput );
      status = DELIQueueRead( inputbuf, (void *)pDelivery, &br, TRUE );
      if( ERR_SUCCESS != status )
      {
                return( status );
      }
      else if( DELIVERY_SHUTDOWN == pDelivery->o_carrier_id )
      {
                bDone = TRUE;
                continue;
      }
#else
      if( 0 == ReadFile( hPipeInputRead, pDelivery, sizeof(
DeliveryDataInput ),
                                        &br, NULL )) {
                status = GetLastError( );
                DELIErrorMessage( ERR_DELIVERY_PIPE_READ );
                continue;
      }
#endif
    }
#ifdef USE_PROCESSES
```

```
#else
   else {
    if(BUF_SUCCESS != bufread( &pDelivery, sizeof( pDeliveryData ),
&br,
                                                INFINITE, inputbuf )) {
            DELIErrorMessage( ERR_DELIVERY_PIPE_READ );
            continue;
    }
   }
#endif
   if( gbDirectConnect )
    retcode = TPCCDeliveryDB( DBC, pDelivery );
   else
    retcode = TPCCDeliveryDeferred( &pDelivery );

   retcode = TPCCDeliveryDeferredResponse( retcode, pDelivery );
  }

  return( status );
}



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

  if ( ERR_DB_PENDING == retcode )
  {
   return( retcode );
  }
  else
  {
   if( ERR_DB_SUCCESS != retcode)
   {

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



   DeltaTime( pDelivery->delta_time );


   if( gbWriteDeliLog )
    DELILog( pDelivery );

   if( gbDisplayCompletions ) {

#ifdef USE_PROCESSES
    status = DELIQueueWrite( outputbuf, (void *)pDelivery,
                                          sizeof( *pDelivery ), TRUE );
    if( ERR_SUCCESS != status )
    {
            DELIErrorMessage(
ERR_DELIVERY_OUTPUT_PIPE_WRITE );
            return( retcode );
    }

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

    UnreserveTransactionStruct( DELIVERY_TRANS, &pDelivery );
   }
  }

  return( retcode );
}
```

deli_srv.h

```
#ifndef DELI_SRV_H
#define DELI_SRV_H
/*+************************************************************
**************
 *                                                            *
 * COPYRIGHT (c) 1997, 2000 BY                                *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 * 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
DIGITAL EQUIPMENT   *
 * CORPORATION.                                               *
 *                                                            *
 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS   *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                            *
 *                                                            *

************************************************************
*************-*/

/*+
 * 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 _WIN32
# define StartTime(num) ((num) = GetTickCount( ))
# define DeltaTime(num) ((num) = GetTickCount( ) - num)
#else
# include <sys/timeb.h>
  typedef int BUFPTR;
# define StartTime(num) \
  {\
   struct timeb tb;\
   ftime( &tb );\
   \
   (num) = ((tb.time % (24*60*60)) * 1000) + tb.millitm;\
  }
# define DeltaTime(num) \
  {\
   struct timeb tb;\
   int now;\
   ftime( &tb );\
   \
   now = ((tb.time % (24*60*60)) * 1000) + tb.millitm;\
   if( now < (num) )\
   {\
    \
    now += (24*60*60*1000);\
```

```
    }\
    (num) = now - (num);\
  }
#endif

#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
# ifdef _WIN32

DELI_GLOBAL(HANDLE
ghPipeInputWrite,INVALID_HANDLE_VALUE);
DELI_GLOBAL(BUFPTR inputbuf,NULL);
DELI_GLOBAL(BUFPTR outputbuf,NULL);
# else
DELI_GLOBAL(BUFPTR inputbuf,-1);
DELI_GLOBAL(BUFPTR outputbuf,-1);
# endif
#endif

int DELIServerStartup( pDeliveryTransportData pDeliveryTransport,
                       pLoginData pLogin, int loginDelay, char
*Path );
int DELIServerShutdown( void );
int DELIQueueRead( int pipe, void *data, unsigned long *plen, BOOL block
);
int DELIQueueWrite( int pipe, void *data, unsigned long len, BOOL block
);

#endif
```

jacket.c

```
#ifdef _WIN32
# include <windows.h>
# include <process.h>
#else
# define __stdcall
# define __cdecl
#endif
#include <time.h>
#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <malloc.h>
#include <errno.h>

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



#define MAXUSERDEF 10


int xact_type;

typedef struct __MyStruct {
  int Termid;
  unsigned Threadid;
  int spid;
  pStartupData pStartup;
} _MyStruct;

_MyStruct *StructArray;
#ifdef _WIN32
HANDLE *HndlArray;
#endif

unsigned __stdcall
ChildRoutine(void * Struct)
{
  int status;
  _MyStruct *pMyStruct;
  CallersContext CC;
  pLoginData          pLogin;
  pNewOrderData pNewOrder;
  pDeliveryData pDelivery;
  pDeliveryData
              pCompletedDeliveries[DELIVERY_RESPONSE_COUNT];
  int                 w_id;
  int                 ld_id;
  int                 ii;
  int                 jj;

  pMyStruct = (_MyStruct *) Struct;


  w_id = (pMyStruct->Termid+1 + 9) / 10;
  ld_id = ((pMyStruct->Termid+1) % 10);

  pLogin = ReserveTransactionStruct( CONNECT_TRANS );
  memcpy( pLogin, &pMyStruct->pStartup->Login, sizeof( *pLogin ));

  status = TPCCConnect( &pLogin );
  CC = pLogin->CC;
  UnreserveTransactionStruct( CONNECT_TRANS, &pLogin );

#if 0
  pNewOrder = ReserveTransactionStruct( NEW_ORDER_TRANS );
  pNewOrder->CC = CC;
  pNewOrder->w_id = w_id;
  pNewOrder->ld_id = ld_id;
  pNewOrder->d_id = ld_id;
  pNewOrder->o_ol_cnt = ((pMyStruct->Termid) % 5) + 10;

  status = TPCCNewOrder( &pNewOrder );
#endif

  for( jj = 0; jj < 5; jj++ ) {
    pDelivery = ReserveTransactionStruct( DELIVERY_TRANS );
    pDelivery->CC = CC;
    pDelivery->w_id = w_id;
    pDelivery->ld_id = ld_id;

    pDelivery->queue_time = 0;
    pDelivery->delta_time = 0;
    pDelivery->o_carrier_id = ((pMyStruct->Termid+1) + 9) / 10;

    status = TPCCDelivery( &pDelivery, pCompletedDeliveries );

    if (status < 0) {
      printf(" Transport error, status = %d. \n", status );
    }
    else if (status > 0) {
      printf(" Database error, status = %d. \n", status );
    }
    else{
#if 0
      printf("The following was found in NewOrder\n");
      printf("Termid = %d Lastname: %s total_amount= %f i_price = %f\n",
                  pMyStruct->Termid, pNewOrder->c_last,
                  pNewOrder->total_amount, pNewOrder->o_ol[0].i_price);

      UnreserveTransactionStruct( NEW_ORDER_TRANS, &pNewOrder );
#else
      printf("The following was found in Delivery\n");
      printf("Termid = %d, "
                  "queue_time = %d, delta_time = %d\n",
                  pMyStruct->Termid,
                  pDelivery->queue_time,
                  pDelivery->delta_time );

      UnreserveTransactionStruct( DELIVERY_TRANS, &pDelivery );

      for( ii = 0;
                  ii < DELIVERY_RESPONSE_COUNT && NULL !=
pCompletedDeliveries[ii];
                  ii++ )
                {
                  printf("\tThe following was found in CompletedDeliveries\n
");
                  printf("\tTermid = %d, "
```

```
                        "queue_time = %d, delta_time = %d, o_id[0] =
%d\n",
                        pMyStruct->Termid,
                        pCompletedDeliveries[ii]->queue_time,
                        pCompletedDeliveries[ii]->delta_time,
                        pCompletedDeliveries[ii]->o_id[0] );

            UnreserveTransactionStruct( DELIVERY_TRANS,
&pCompletedDeliveries[ii] );

          }
#endif
   }
   sleep( 2 );
  }



  return(0);
}



int __cdecl
main(int argc, char *argv[])
{

  unsigned long ulHandle;
  int status;
  int NumChild, TotalUsers = 1;
  StartupData           Startup;
  pStartupData          pStartup = &Startup;

  if (argc != 2)
   {
    printf("Usage: %s <#users>\n", argv[0] );
    printf("If #users not specified %d is assumed.\n", TotalUsers);
   }
  else
   TotalUsers = atoi(argv[1]);
  StructArray = (_MyStruct *) malloc(sizeof(_MyStruct)*TotalUsers);
#ifdef _WIN32
  HndlArray = (HANDLE *) malloc(sizeof(HANDLE) * TotalUsers);
#endif

  pStartup->Config.maxConnections = TotalUsers;
  pStartup->Config.maxDistricts = TotalUsers;

  pStartup->DeliveryTransport.display_completions = TRUE;
  pStartup->DeliveryTransport.flush_log = FALSE;
  pStartup->DeliveryTransport.num_threads = 1;
  pStartup->DeliveryTransport.num_queued_deliveries = 10;
  pStartup->DeliveryTransport.num_queued_responses = 10;
  pStartup->DeliveryTransport.use_transport = FALSE;
  pStartup->DeliveryTransport.use_pipe = FALSE;
  pStartup->DeliveryTransport.write_log = FALSE;

  pStartup->Transport.asynchronous = FALSE;
  pStartup->Transport.generic = TRUE;
  pStartup->Transport.num_gc = 1;
  pStartup->Transport.num_dy = 0;
  pStartup->Transport.num_no = 0;
  pStartup->Transport.num_os = 0;
  pStartup->Transport.num_pt = 0;
  pStartup->Transport.num_sl = 0;

  strcpy( pStartup->Login.databaseLogin.szServer, "csg239" );
  strcpy( pStartup->Login.databaseLogin.szDatabase, "tpcc" );
  strcpy( pStartup->Login.databaseLogin.szUser, "sa" );
  strcpy( pStartup->Login.databaseLogin.szPassword, "" );
  strcpy( pStartup->Login.databaseLogin.szApplication, "app" );

  pStartup->loginDelay = 0;
  pStartup->Path = ".";

  pStartup->pTPCCNewOrderResponse = NULL;
  pStartup->pTPCCOrderStatusResponse = NULL;
  pStartup->pTPCCPaymentResponse = NULL;
  pStartup->pTPCCStockLevelResponse = NULL;
  pStartup->pTPCCResponseComplete = NULL;

  status = TPCCOpenLog( pStartup->Path );
  status = TPCCStartup( pStartup );

  for(NumChild=0; NumChild < TotalUsers; NumChild++)
  {
   StructArray[NumChild].pStartup = pStartup;
   StructArray[NumChild].Termid = NumChild;
#ifdef _WIN32
   ulHandle =  _beginthreadex(NULL, 0, ChildRoutine,
                                        (void *)
&StructArray[NumChild],

0,&StructArray[NumChild].Threadid);
   if (ulHandle <= 0)
   {
    fprintf(stderr, "Begin Thread Failed. %d \n",errno);
    exit(errno);
   }
   else
   {
    HndlArray[NumChild] = (HANDLE) ulHandle;
   }
#else
   ChildRoutine( &StructArray[NumChild] );
#endif

  }
#ifdef _WIN32
  SleepEx(INFINITE, TRUE);
#else
  select( 0, NULL, NULL, NULL, NULL );
#endif
  return 0;
}
```

logfile.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                          *
 *                                                          *

**********************************************************
************-*/

/*+
 * Abstract: This file contains the Digital created front end functions
 *                              for the tpcc benchmark.
 *
```

```
 * Author: W Carr
 * Creation Date: October 1997
 *
 * Modified history:
 *
 *
*/

#ifdef _WIN32
# include <windows.h>
#else
# include <sys/time.h>
# include <stdarg.h>
#endif
#include <stdio.h>

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


static char             LogPath[FILENAMESIZE];

#ifdef _WIN32
static HANDLE           hLogFile     =
INVALID_HANDLE_VALUE;

static CRITICAL_SECTION ErrCriticalSection;
static CRITICAL_SECTION LogCriticalSection;

static BOOL             CritSecInitialized = FALSE;
#else
static FILE             *hLogFile    = NULL;
#endif


BOOL
TPCCOpenLog( char *Path )
{
  char                  File[FILENAMESIZE];

  strcpy( LogPath, Path );

#ifdef _WIN32
  InitializeCriticalSection( &LogCriticalSection );
  InitializeCriticalSection( &ErrCriticalSection );
  CritSecInitialized = TRUE;
#endif

  strcpy( File, LogPath );
  strcat( File, "tpcclog" );

#ifdef _WIN32
  hLogFile = CreateFile( File, GENERIC_WRITE, FILE_SHARE_READ,
NULL,
                         CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL );
#else
  hLogFile = fopen( File, "w" );
#endif

  strcpy( File, LogPath );
  strcat( File, "tpccerr" );
  unlink( File );

#ifdef _WIN32
  return( INVALID_HANDLE_VALUE != hLogFile );
#else
  return( NULL != hLogFile );
#endif
}


BOOL
TPCCCloseLog( void )
{
#ifdef _WIN32
  CloseHandle( hLogFile );

  DeleteCriticalSection( &LogCriticalSection );
  DeleteCriticalSection( &ErrCriticalSection );
#else
  fclose( hLogFile );
#endif

  return TRUE;
}


void
TPCCLog( char *fmt, ... )
{
  va_list               marker;
  char                  szArg[4096];
  int                   len;
#ifdef _WIN32
  char                  szTmp[4096];
  SYSTEMTIME            systemTime;
  DWORD                 dwWriteLen;
#else
  struct timeval        current;
  char                  *dayTime;
  time_t                sec;
#endif

  va_start( marker, fmt );
  vsprintf( szArg, fmt, marker );
  va_end( marker );

#ifdef _WIN32
  GetLocalTime( &systemTime );

  len = sprintf( szTmp,
                 "%2.2d/%2.2d/%2.2d
%2.2d:%2.2d:%2.2d\t%s\r\n",
                 systemTime.wYear, systemTime.wMonth,
systemTime.wDay,
                 systemTime.wHour, systemTime.wMinute,
systemTime.wSecond,
                 szArg );

  EnterCriticalSection( &LogCriticalSection );
  WriteFile( hLogFile, szTmp, len, &dwWriteLen, NULL );
  LeaveCriticalSection( &LogCriticalSection );
#else
  gettimeofday( &current, NULL );
  sec = current.tv_sec;
  dayTime = ctime( &sec );
  dayTime[28] = '\0';

  fprintf( hLogFile, "%s\t%s\n", dayTime, szArg );
#endif
}

int
TPCCErrInternal( char *szTmp, int len )
{
  char                  File[FILENAMESIZE];
#ifdef _WIN32
  DWORD                 dwWriteLen;
  HANDLE                hErrFile;
#else
  FILE                  *hErrFile;
#endif

  strcpy( File, LogPath );
  strcat( File, "tpccerr" );

#ifdef _WIN32
  if( !CritSecInitialized )
    return( ERR_LOGFILE_SUBSYSTEM_NOT_INITED );

  EnterCriticalSection( &ErrCriticalSection );

  hErrFile = CreateFile( File, GENERIC_WRITE, FILE_SHARE_READ,
NULL,
                         OPEN_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL );
  SetFilePointer( hErrFile, 0, 0, FILE_END );
```

```
  WriteFile( hErrFile, szTmp, len, &dwWriteLen, NULL );

  CloseHandle( hErrFile );

  LeaveCriticalSection( &ErrCriticalSection );
#else
  hErrFile = fopen( File, "a" );

  fwrite( szTmp, len, 1, hErrFile );

  fclose( hErrFile );
#endif

  return( ERR_SUCCESS );
}


void
TPCCErr( char *fmt, ...)
{
  va_list                 marker;
  char                    szArg[4096];
  int                     len;
  char                    szTmp[4096];
#ifdef _WIN32
  SYSTEMTIME              systemTime;
#else
  struct timeval          current;
  char                    *dayTime;
  time_t                  sec;
#endif

  va_start( marker, fmt );
  vsprintf( szArg, fmt, marker );
  va_end( marker );

#ifdef _WIN32
  GetLocalTime( &systemTime );

  len = sprintf( szTmp,
                 "%2.2d/%2.2d/%2.2d
%2.2d:%2.2d:%2.2d\t%s\r\n",
                 systemTime.wYear, systemTime.wMonth,
systemTime.wDay,
                 systemTime.wHour, systemTime.wMinute,
systemTime.wSecond,
                 szArg );
#else
  gettimeofday( &current, NULL );
  sec = current.tv_sec;
  dayTime = ctime( &sec );
  dayTime[28] = '\0';

  len = sprintf( szTmp, "%s\t%s\n", dayTime, szArg );
#endif

  TPCCErrInternal( szTmp, len );
}
```

reg.c

```
/*+**********************************************************
***************
 *                                                          *
 *  COPYRIGHT (c) 1999, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
************-*/

/*+
 * Abstract: This is the source file for registry manipulation functions.
 *
 * Author: WCarr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
-*/


#include <windows.h>

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

int
GetConfigValue( char *Name, int *Len, char *Value )
{
  int                             Status;

  Status = GetRegValue( TPCC_CLASS, Name, Len, Value );

  return( Status );
}

int
GetRegValue( char *Class, char *Name, int *Len, char *Value )
{
  HKEY                            hKey;
  int                             Status;
  int                             Type;

  Status = RegOpenKeyEx( HKEY_LOCAL_MACHINE, Class, 0,
KEY_READ, &hKey );
  if( ERROR_SUCCESS != Status )
  {
    Status = ERR_CANT_FIND_CLASS;
  }
  else
  {
    Status = RegQueryValueEx( hKey, Name, 0, &Type, Value, Len );
    if ( ERROR_SUCCESS != Status )
    {
      Status = ERR_CANT_FIND_VALUE;
    }
    else
    {
      Status = ERR_SUCCESS;
    }

    RegCloseKey( hKey );
  }

  return( Status );
}

int
```

```
SetConfigValue( char *Name, char *Value )
{
 int                                    Status;

 Status = SetRegValue( TPCC_CLASS, Name, Value );

 return( Status );
}

int
SetRegValue( char *Class, char *Name, char *Value )
{
 DWORD                                  dwDisposition;
 HKEY                                   hKey;
 DWORD                                  Len;
 int                                    Status;
 int                                    Type;

 Status = RegCreateKeyEx(HKEY_LOCAL_MACHINE, Class,
                                        0, NULL,
REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS,
                                        NULL, &hKey, &dwDisposition);
 if( ERROR_SUCCESS != Status )
 {
  return( ERR_CANT_CREATE_CONFIG_DATABASE );
 }
 else
 {
  Len = strlen( Value );
  Type = REG_SZ;
  Status = RegSetValueEx( hKey, Name, 0, Type, Value, Len );
  if ( ERROR_SUCCESS != Status )
  {
   Status = ERR_CANT_SET_CONFIG_VALUE;
  }
  else
  {
   Status = ERR_SUCCESS;
  }

  RegCloseKey( hKey );
 }

 return( Status );
}
```

reg.h

```
#ifndef REG_H
#define REG_H

/*+**********************************************************
**************
 *                                                          *
 * COPYRIGHT (c) 1997, 2000 BY                              *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 * 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
DIGITAL EQUIPMENT  *
 * CORPORATION.                                             *
 *                                                          *
 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
************-*/

/*+
 * Abstract: This is the header file for reg.c.
 *
 * Author: WCarr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
 */


#define INETINFO_CLASS \

                "SYSTEM\\CurrentControlSet\\Services\\InetInfo\\Parameters"

#define TPCC_CLASS "SOFTWARE\\Compaq\\TPCC"



int GetRegValue( char *Class, char *Name, int *Len, char *Value );
int SetRegValue( char *Class, char *Name, char *Value );

#endif
```

sybase_db.c

```
/*+**********************************************************
**************
 *                                                          *
 * COPYRIGHT (c) 1997, 2000 BY                              *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 * 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
DIGITAL EQUIPMENT  *
 * CORPORATION.                                             *
 *                                                          *
 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS  *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
************-*/
#ifdef _WIN32
# include <windows.h>
#else
# include <unistd.h>
# define GetCurrentThreadId getpid
#endif
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
```

```
#include <sys/timeb.h>

#if defined _WIN32
# include <crtdbg.h>
#else
# define _ASSERT(arg)
#endif

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

#include <sybfront.h>
#include <sybdb.h>

#define DEFCLPACKSIZE 4096
#define DEADLOCKWAIT  10
#define DEADLOCKRETRIES 3

#define NEW_SP

#define UtilStrCpy(x,y,cnt) (strncpy(x,y,cnt), x[cnt] = '\0')
#define COPYDATE(datedata,daterec) \
datedata.year = daterec.dateyear;\
datedata.month = daterec.datemonth;\
datedata.day = daterec.datedmonth;\
datedata.hour = daterec.datehour;\
datedata.minute = daterec.dateminute;\
datedata.second = daterec.datesecond



#define UtilConvert2Char(string,i) ((10*(string[i]-0x30)) + (string[i+1]-
0x30))

typedef struct _UserData {
  BOOL bDeadlock;
  int iDBStatus;
} UserData;

static UserData gUD;
static UserData *gpUD = &gUD;


int CS_PUBLIC DBFAR syb_err_fnc(DBContext dbproc, int severity, int
dberr, int oserr, char *dberrstr, char *oserrstr );
int CS_PUBLIC DBFAR syb_msg_fnc(DBContext DBFAR dbproc, DBINT
msgno, int msgstate, int severity, char DBFAR *msgtext, char DBFAR
*svrname, char DBFAR *procname, int line);
BOOL SQLDetectDeadlock(DBContext dbproc);
void convert_date_and_time ( DBDateData *dstruct, char *dstring);
void convert_date ( DBDateData *dstruct, char *dstring);


int
TPCCStartupDB( pConfigData pConfig )
{
  dbinit();
  if ( dbgetmaxprocs() < pConfig->maxConnections ) {
    if ( dbsetmaxprocs((short)pConfig->maxConnections) == FAIL ) {



      return ERR_CAN_NOT_SET_MAX_CONNECTIONS;
    }
  }

  dbmsghandle( syb_msg_fnc );
  dberrhandle( syb_err_fnc );


  return ERR_DB_SUCCESS;
}

int
TPCCShutdownDB( void )
{
  dbexit();

  return 0;
}


int CS_PUBLIC DBFAR syb_err_fnc(DBContext dbproc, int severity, int
dberr, int oserr, char *dberrstr, char *oserrstr) {
  UserData                                          *pUD;

  TPCCErr( "dbproc = %8X, Threadid = %d\r\n", dbproc,
GetCurrentThreadId( ));

  if ((dbproc == NULL) || (DBDEAD(dbproc))) {
    TPCCErr( "oserr=%d dberr=%d\r\n%s\r\n%s\r\nAND dbproc is
%s.\r\n\r\n",
                    oserr, dberr, oserrstr, dberrstr,
                    (NULL == dbproc) ? "NULL":"Dead" );
    if( NULL == dbproc )
      return INT_CANCEL;
  }

  if ( !(pUD = (UserData *)dbgetuserdata(dbproc)) ) {
    pUD = gpUD;
  }

  if ( pUD && (DBNOERR != pUD->iDBStatus) )
    return INT_CANCEL;

  if ( oserr != DBNOERR ) {
    if ( pUD )
      pUD->iDBStatus = oserr;

    TPCCErr( "%s\r\n", oserrstr );
  }

  if ( dberr != DBNOERR ) {
    if ( pUD )
      pUD->iDBStatus = dberr;

    TPCCErr( "%s\r\n", dberrstr );
  }

  return INT_CANCEL;
}


int
CS_PUBLIC DBFAR syb_msg_fnc( DBContext DBFAR dbproc, DBINT
msgno, int msgstate,
                                          int severity, char *msgtext, char
*svrname,
                                          char *procname, int line )
{
  UserData    *pUD;

  if ( !(pUD = (UserData *)dbgetuserdata(dbproc)) ) {
    pUD = gpUD;
  }


  if ( (msgno == 5701) || (msgno == 2528) || (msgno == 5703) || (msgno ==
6006) )
    return 0;


  if (msgno == 1205) {

    if ( pUD )
      pUD->bDeadlock = TRUE;
    else {
      TPCCErr( "UserData is NULL\r\n" );
    }
    return 0;
  }
  if ( pUD && (DBNOERR != pUD->iDBStatus) )
    return 0;

  if (msgno == 0)
    return 0;
  else {
    if ( pUD )
      pUD->iDBStatus = msgno;
```

```
    TPCCErr( "Error: SQLSVR(%d): %s\r\n", msgno, msgtext);
  }

  return 0;
}


int
TPCCConnectDB( DBContext *dbproc, pLoginData pLogin )
{
  LOGINREC                *login;
  UserData    *pUD;
  struct timeb   timebuf[10];
  int                             spid = 0;


  ftime(&timebuf[0]);
  login = dblogin();
  if ( !*pLogin->databaseLogin.szUser )
    DBSETLUSER(login, "sa");
  else
    DBSETLUSER(login, pLogin->databaseLogin.szUser);

  DBSETLPWD(login,  pLogin->databaseLogin.szPassword);
  DBSETLHOST(login, pLogin->databaseLogin.szApplication);

  DBSETLPACKET(login, (unsigned short)DEFCLPACKSIZE);

  if ((*dbproc = dbopen(login, pLogin->databaseLogin.szServer )) ==
NULL)
    return ERR_DB_INTERFACE;

  ftime(&timebuf[1]);

  pUD = (UserData *)malloc(sizeof(UserData));
  pUD->bDeadlock = FALSE;
  pUD->iDBStatus = DBNOERR;
  dbsetuserdata(*dbproc, (BYTE *) pUD);
  ftime(&timebuf[2]);


  dbuse(*dbproc, pLogin->databaseLogin.szDatabase);

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

  dbsqlexec(*dbproc);
  ftime(&timebuf[3]);
  while (dbresults(*dbproc) != NO_MORE_RESULTS) {
    dbbind(*dbproc, 1, SMALLBIND, (DBINT) 0, (BYTE *) &spid);
    while (dbnextrow(*dbproc) != NO_MORE_ROWS)
      ;
  }
  ftime(&timebuf[4]);
  dbcmd(*dbproc, "set nocount on");

  dbsqlexec(*dbproc);
  ftime(&timebuf[5]);
  while (dbresults(*dbproc) != NO_MORE_RESULTS) {
    while (dbnextrow(*dbproc) != NO_MORE_ROWS)
      ;
  }
  ftime(&timebuf[6]);
  TPCCLog( "TPCCConnectDB times: "
                  "%d.%03d %d.%03d %d.%03d %d.%03d "
                  "%d.%03d %d.%03d %d.%03d \r\n",
                  timebuf[0].time,timebuf[0].millitm,
                  timebuf[1].time,timebuf[1].millitm,
                  timebuf[2].time,timebuf[2].millitm,
                  timebuf[3].time,timebuf[3].millitm,
                  timebuf[4].time,timebuf[4].millitm,
                  timebuf[5].time,timebuf[5].millitm,
                  timebuf[6].time,timebuf[6].millitm );
  return ERR_DB_SUCCESS;
}


int
TPCCDisconnectDB( DBContext dbproc, pConnData pConn ) {
  UserData    *pUD;

  if ( dbproc ) {
    if ( (pUD = (UserData *)dbgetuserdata(dbproc)) ) {
      free(pUD);
    }

    dbclose(dbproc);
  }
  return ERR_DB_SUCCESS;
}

int
TPCCStockLevelDB( DBContext dbproc, pStockLevelData pStockLevel )
{
  int                             tryit;
  RETCODErc;
  UserData    *pUD;
  int                             temp_iid, i, count, uniq[500];
  int                             Status;

  TRANSACTION_DEBUG_STAGE( pStockLevel, IN_DB );

  Status = ERR_DB_DEADLOCK_LIMIT;


  if ( (pUD = (UserData *)dbgetuserdata(dbproc)) ) {
    pUD->iDBStatus = DBNOERR;
  }

  for (tryit=0; tryit < DEADLOCKRETRIES; tryit++) {
    if (dbrpcinit(dbproc, "stock_level", 0) == SUCCEED)  {
      dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, (BYTE
*)&pStockLevel->w_id);
      dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, (BYTE
*)&pStockLevel->ld_id);
      dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,
                              (BYTE *)&pStockLevel->threshold);

      if (dbrpcsend(dbproc) == SUCCEED && dbsqlok(dbproc) ==
SUCCEED) {
                  if (rc = dbresults(dbproc) == SUCCEED) {
#ifdef NEW_SP
                    dbbind(dbproc, 1, INTBIND, 0, (BYTE *) &temp_iid);
                    count = 0;
                    while (dbnextrow(dbproc) == REG_ROW) {
                      for (i = 0; i < count; i++)
                        if (temp_iid == uniq[i]) break;
                      if (i >= 500) {
                        Status = ERR_DB_ERROR;
                        break;
                      }
                      else uniq[count++] = temp_iid;
                    }
                    dbcanquery(dbproc);
                    pStockLevel->low_stock = count;
#else
                    dbbind(dbproc, 1, INTBIND, 0, (BYTE *) &pStockLevel-
>low_stock);
                    if (dbnextrow(dbproc) == REG_ROW)
                      dbcanquery(dbproc);
#endif
                  }
      }
      else {
                  return ERR_DB_ERROR;
      }
    }

    if (SQLDetectDeadlock(dbproc)) {
      Sleep(DEADLOCKWAIT*tryit);
    }
    else {
      if( DBNOERR != pUD->iDBStatus ) {
                  Status = ERR_DB_ERROR;
                  break;
      }
      else {
                  Status = ERR_DB_SUCCESS;
                  break;
      }
```

```
  }
 }
 TRANSACTION_DEBUG_STAGE( pStockLevel, LEAVING_DB );


 return( Status );
}



#pragma message ("FIXME: return code is overloaded.  How to report
invalid item number?")
int
TPCCNewOrderDB( DBContext dbproc, pNewOrderData pNewOrder )
{
 RETCODE                rc;
 int                            i;
 DBINT                          commit_flag = 1;
 int                            tryit;
 char                           tmpbuf[20];
 UserData               *pUD;
 int                            always_zero=0;
 DBDATEREC                      tmp_daterec;
 DBDATETIME                     tmp_datetime;
 int                            Status;
#ifdef _DEBUG
 int                    num_cols;
#endif

 TRANSACTION_DEBUG_STAGE( pNewOrder, IN_DB );

 Status = ERR_DB_DEADLOCK_LIMIT;

 if ( (pUD = (UserData *)dbgetuserdata(dbproc)) ) {
  pUD->iDBStatus = DBNOERR;
 }

 if (pNewOrder->o_all_local)
  strncpy(tmpbuf, "neworder_local", 15);
 else
  strncpy(tmpbuf, "neworder_remote", 16);

 for (tryit=0; tryit < DEADLOCKRETRIES; tryit++) {
  if (dbrpcinit(dbproc, tmpbuf, 0) == SUCCEED)  {
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, (BYTE *)
&pNewOrder->w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, (BYTE *)
&pNewOrder->d_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, (BYTE *)
&pNewOrder->c_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, (BYTE *)
&pNewOrder->o_ol_cnt);

#ifndef NEW_SP

    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, (BYTE *)
&always_zero);

    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, (BYTE *)
&pNewOrder->o_ol_cnt);

    dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, (BYTE *)
&pNewOrder->o_id);
    dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1, (BYTE *)
&pNewOrder->tax_n_discount );
    dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1, (BYTE *)
&commit_flag);
#endif
    for (i = 0; i < pNewOrder->o_ol_cnt; i++) {
            dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1,
                        (BYTE *) &pNewOrder->o_ol[i].ol_i_id);
            if (! pNewOrder->o_all_local)
              dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,
                        (BYTE *) &pNewOrder-
>o_ol[i].ol_supply_w_id);
            dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,
                        (BYTE *) &pNewOrder->o_ol[i].ol_quantity);
    }

    if (dbrpcsend(dbproc) == SUCCEED && dbsqlok(dbproc) ==
SUCCEED) {
            pNewOrder->total_amount=0;


            for (i = 0; i < pNewOrder->o_ol_cnt; i++) {
              if (dbresults(dbproc)  == SUCCEED)  {
                dbbind(dbproc, 1, NTBSTRINGBIND, sizeof(pNewOrder-
>o_ol[i].i_name),
                        pNewOrder->o_ol[i].i_name);
                dbbind(dbproc, 2, INTBIND, 0,
                        (BYTE *) &pNewOrder->o_ol[i].s_quantity);
                dbbind(dbproc, 3, FLT8BIND, 0,
                        (BYTE *) &pNewOrder->o_ol[i].i_price);
                dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(pNewOrder-
>o_ol[i].b_g),
                        pNewOrder->o_ol[i].b_g);

                if (dbnextrow(dbproc) == REG_ROW) {

                  pNewOrder->o_ol[i].ol_amount =
                        pNewOrder->o_ol[i].i_price * pNewOrder-
>o_ol[i].ol_quantity;
                  pNewOrder->total_amount += pNewOrder-
>o_ol[i].ol_amount;

                  if (pNewOrder->o_ol[i].i_name[0] == '\0') commit_flag = 0;
                 }
                 rc = dbcanquery(dbproc);
                }
               }


            if (dbresults(dbproc) == SUCCEED) {
              _ASSERT( 7 == ( num_cols = dbnumcols( dbproc )));
              dbbind(dbproc, 1, FLT8BIND, 0, (BYTE *) &pNewOrder-
>w_tax);
              dbbind(dbproc, 2, FLT8BIND, 0, (BYTE *) &pNewOrder-
>d_tax);
              dbbind(dbproc, 3, INTBIND,  0, (BYTE *) &pNewOrder-
>o_id);
              dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(pNewOrder-
>c_last),
                        pNewOrder->c_last);
              dbbind(dbproc, 5, FLT8BIND, 0, (BYTE *) &pNewOrder-
>c_discount);
              dbbind(dbproc, 6, NTBSTRINGBIND, sizeof(pNewOrder-
>c_credit),
                        pNewOrder->c_credit);
#ifdef NEW_SP
              dbbind(dbproc, 7, DATETIMEBIND, 0, (BYTE *)
&tmp_datetime);
#else
              dbbind(dbproc, 7, FLT8BIND, 0, (BYTE *) &pNewOrder-
>tax_n_discount);
              dbbind(dbproc, 8, NTBSTRINGBIND, sizeof(tmpbuf),
tmpbuf);
#endif
              if (dbnextrow(dbproc) == REG_ROW ) {
                dbcanquery(dbproc);
#ifdef NEW_SP
                dbdatecrack (NULL, &tmp_daterec, &tmp_datetime);
                COPYDATE(pNewOrder->o_entry_d, tmp_daterec);
                pNewOrder->tax_n_discount = (1.0 - pNewOrder-
>c_discount) *
                  (1.0 + pNewOrder->w_tax + pNewOrder->d_tax);
#else
                convert_date_and_time(&pNewOrder->o_entry_d, tmpbuf);
#endif
              }
             }
    }
    else {
            return ERR_DB_ERROR;
    }
   }
   if (SQLDetectDeadlock(dbproc)) {
    Sleep(DEADLOCKWAIT*tryit);
   }
   else {
```

```
    if (commit_flag == 1) {
            pNewOrder->total_amount *= pNewOrder->tax_n_discount;
            if( DBNOERR != pUD->iDBStatus ) {
              Status = ERR_DB_ERROR;
              break;
            }
            else {
              Status = ERR_DB_SUCCESS;
              break;
            }
    }
    else {
            Status =  ERR_DB_NOT_COMMITED;
            break;
    }
   }
  }
  TRANSACTION_DEBUG_STAGE( pNewOrder, LEAVING_DB );


  return( Status );
}



int
TPCCPaymentDB( DBContext dbproc, pPaymentData pPayment )
{
  int                   tryit;
  char                  tmpbuf[20];
  BOOL                  by_name;
  UserData    *pUD;
  DBDATEREC             tmp_daterec;
  DBDATETIME            tmp_date, tmp_datetime;
  int                   Status;
#ifdef _DEBUG
  int                   num_cols;
#endif

  TRANSACTION_DEBUG_STAGE( pPayment, IN_DB );

  Status = ERR_DB_DEADLOCK_LIMIT;

  if ( (pUD = (UserData *)dbgetuserdata(dbproc)) ) {
    pUD->iDBStatus = DBNOERR;
  }

  if (pPayment->c_id == INVALID_C_ID) {
    strncpy(tmpbuf, "payment_byname", 15);
    by_name = TRUE;
  }
  else {
    strncpy(tmpbuf, "payment_byid", 13);
    by_name = FALSE;
  }

  for (tryit=0; tryit < DEADLOCKRETRIES; tryit++) {
    if (dbrpcinit(dbproc, tmpbuf, 0) == SUCCEED)  {
      dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,(BYTE *)&pPayment-
>w_id);
      dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,(BYTE *)&pPayment-
>c_w_id);
      dbrpcparam(dbproc, NULL, 0, SYBFLT8, -1, -1,(BYTE *)&pPayment-
>h_amount);
      dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1,(BYTE *)&pPayment-
>d_id);
      dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1,(BYTE *)&pPayment-
>c_d_id);
      if (by_name == TRUE)
            dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1,
strlen(pPayment->c_last),
                        pPayment->c_last);
      else
            dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1, (BYTE *)
&pPayment->c_id);

      if (dbrpcsend(dbproc) == SUCCEED && dbsqlok(dbproc) ==
SUCCEED
            && dbresults(dbproc) == SUCCEED) {
            _ASSERT( 27 == ( num_cols = dbnumcols( dbproc )));
      dbbind(dbproc,  1, INTBIND,     0, (BYTE *) &pPayment->c_id);
      dbbind(dbproc,  2, NTBSTRINGBIND, sizeof(pPayment->c_last),
                  pPayment->c_last);
#ifdef NEW_SP
            dbbind(dbproc,  3, DATETIMEBIND,  0, (BYTE *)
&tmp_datetime);
#else
      dbbind(dbproc,  3, NTBSTRINGBIND, sizeof(tmpbuf), tmpbuf);
#endif

      dbbind(dbproc,  4, NTBSTRINGBIND, sizeof(pPayment->w_street_1),
                  pPayment->w_street_1);
      dbbind(dbproc,  5, NTBSTRINGBIND, sizeof(pPayment->w_street_2),
                  pPayment->w_street_2);
      dbbind(dbproc,  6, NTBSTRINGBIND, sizeof(pPayment->w_city),
                  pPayment->w_city);
      dbbind(dbproc,  7, NTBSTRINGBIND, sizeof(pPayment->w_state),
                  pPayment->w_state);
      dbbind(dbproc,  8, NTBSTRINGBIND, sizeof(pPayment->w_zip),
                  pPayment->w_zip);

      dbbind(dbproc,  9, NTBSTRINGBIND, sizeof(pPayment->d_street_1),
                  pPayment->d_street_1);
      dbbind(dbproc, 10, NTBSTRINGBIND, sizeof(pPayment->d_street_2),
                  pPayment->d_street_2);
      dbbind(dbproc, 11, NTBSTRINGBIND, sizeof(pPayment->d_city),
                  pPayment->d_city);
      dbbind(dbproc, 12, NTBSTRINGBIND, sizeof(pPayment->d_state),
                  pPayment->d_state);
      dbbind(dbproc, 13, NTBSTRINGBIND, sizeof(pPayment->d_zip),
                  pPayment->d_zip);

      dbbind(dbproc, 14, NTBSTRINGBIND, sizeof(pPayment->c_first),
                  pPayment->c_first);
      dbbind(dbproc, 15, NTBSTRINGBIND, sizeof(pPayment->c_middle),
                  pPayment->c_middle);
      dbbind(dbproc, 16, NTBSTRINGBIND, sizeof(pPayment->c_street_1),
                  pPayment->c_street_1);
      dbbind(dbproc, 17, NTBSTRINGBIND, sizeof(pPayment->c_street_2),
                  pPayment->c_street_2);
      dbbind(dbproc, 18, NTBSTRINGBIND, sizeof(pPayment->c_city),
                  pPayment->c_city);
      dbbind(dbproc, 19, NTBSTRINGBIND, sizeof(pPayment->c_state),
                  pPayment->c_state);
      dbbind(dbproc, 20, NTBSTRINGBIND, sizeof(pPayment->c_zip),
                  pPayment->c_zip);
      dbbind(dbproc, 21, NTBSTRINGBIND, sizeof(pPayment->c_phone),
                  pPayment->c_phone);
#ifdef NEW_SP
            dbbind(dbproc,  22, DATETIMEBIND,  0, (BYTE *)
&tmp_date);
#else
      dbbind(dbproc, 22, NTBSTRINGBIND, sizeof(tmpbuf2), tmpbuf2);
#endif
      dbbind(dbproc, 23, NTBSTRINGBIND, sizeof(pPayment->c_credit),
                  pPayment->c_credit);
      dbbind(dbproc, 24, FLT8BIND,      0, (BYTE *) &pPayment-
>c_credit_lim);
      dbbind(dbproc, 25, FLT8BIND,      0, (BYTE *) &pPayment-
>c_discount);
      dbbind(dbproc, 26, FLT8BIND,      0, (BYTE *) &pPayment-
>c_balance);
      dbbind(dbproc, 27, NTBSTRINGBIND, sizeof(pPayment->c_data),
                  pPayment->c_data);

      if (dbnextrow(dbproc) == REG_ROW ) {
            dbcanquery(dbproc);
#ifdef NEW_SP
            dbdatecrack(NULL, &tmp_daterec, &tmp_datetime);
            COPYDATE(pPayment->h_date, tmp_daterec);
            dbdatecrack(NULL, &tmp_daterec, &tmp_date);
            COPYDATE(pPayment->c_since, tmp_daterec);

#else
            convert_date_and_time(&pPayment->h_date, tmpbuf);
            convert_date(&pPayment->c_since, tmpbuf2);
#endif
          }
    }
    else {
```

```
                return ERR_DB_ERROR;
      }
    }
    if (SQLDetectDeadlock(dbproc)) {
      Sleep(DEADLOCKWAIT*tryit);
    }
    else {
      if ( pPayment->c_id == INVALID_C_ID ) {
#pragma message ("FRANCOIS: How is testing customer ID a complete test
of success of the transaction?")
                Status = ERR_DB_NOT_COMMITED;
                break;
      }
      else {
                if( DBNOERR != pUD->iDBStatus ) {
                  Status = ERR_DB_ERROR;
                  break;
                }
                else {
                  Status =  ERR_DB_SUCCESS;
                  break;
                }
      }
    }
  }
  TRANSACTION_DEBUG_STAGE( pPayment, LEAVING_DB );


  return( Status );
}


int
TPCCOrderStatusDB( DBContext dbproc, pOrderStatusData pOrderStatus )
{
  int                   tryit;
  int                   i;
  char                  tmpbuf[20];
  BOOL                  by_name;
  UserData    *pUD;
  struct status_order_line temp_ol;
  DBDATEREC             tmp_daterec;
  DBDATETIME            tmp_datetime;
  int                   Status;
#ifdef _DEBUG
  int                   num_cols;
#endif

  TRANSACTION_DEBUG_STAGE( pOrderStatus, IN_DB );

  Status = ERR_DB_DEADLOCK_LIMIT;

  if ( (pUD = (UserData *)dbgetuserdata(dbproc)) ) {
    pUD->iDBStatus = DBNOERR;
  }

  if (pOrderStatus->c_id == INVALID_C_ID) {
    strncpy(tmpbuf, "order_status_byname", 20);
    by_name = TRUE;
  }
  else {
    strncpy(tmpbuf, "order_status_byid", 18);
    by_name = FALSE;
  }

  for (tryit=0; tryit < DEADLOCKRETRIES; tryit++) {
    if (dbrpcinit(dbproc, tmpbuf, 0) == SUCCEED)  {
      dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,(BYTE
*)&pOrderStatus->w_id);
      dbrpcparam(dbproc, NULL, 0, SYBINT1, -1, -1,(BYTE
*)&pOrderStatus->d_id);
      if (by_name == TRUE)
                dbrpcparam(dbproc, NULL, 0, SYBCHAR, -1,
strlen(pOrderStatus->c_last),
                          pOrderStatus->c_last);
      else
                dbrpcparam(dbproc, NULL, 0, SYBINT4, -1, -1,
                          (BYTE *) &pOrderStatus->c_id);


    }
    if (dbrpcsend(dbproc) == SUCCEED && dbsqlok(dbproc) == SUCCEED
&&
                dbresults(dbproc) == SUCCEED) {
#if 0
                _ASSERT( 5 == ( num_cols = dbnumcols( dbproc )));
#endif

      dbbind(dbproc, 1, INTBIND,     0,     (BYTE *)
&temp_ol.ol_supply_w_id);
      dbbind(dbproc, 2, INTBIND,     0,     (BYTE *) &temp_ol.ol_i_id);
      dbbind(dbproc, 3, INTBIND,     0,     (BYTE *) &temp_ol.ol_quantity);
      dbbind(dbproc, 4, FLT8BIND,    0,     (BYTE *) &temp_ol.ol_amount);
#ifdef NEW_SP
      dbbind(dbproc, 5, DATETIMEBIND,0, (BYTE *) &tmp_datetime);
#else
      dbbind(dbproc, 5, NTBSTRINGBIND, sizeof(tmpbuf),tmpbuf);
#endif

      for (i = 0; (dbnextrow(dbproc)) == REG_ROW; i++) {

                memcpy(&pOrderStatus->s_ol[i], &temp_ol,
                     sizeof(struct status_order_line) - sizeof(DBDateData));
#ifdef NEW_SP
                dbdatecrack( NULL, &tmp_daterec, &tmp_datetime);
                COPYDATE(pOrderStatus->s_ol[i].ol_delivery_d,
tmp_daterec);
#else
                convert_date(&pOrderStatus->s_ol[i].ol_delivery_d, tmpbuf);
#endif
      }
      pOrderStatus->o_ol_cnt = i ;


      if (dbresults(dbproc) == SUCCEED) {
                _ASSERT( 8 == ( num_cols = dbnumcols( dbproc )));
                dbbind(dbproc, 1, INTBIND,       0,     (BYTE *)
&pOrderStatus->c_id);
                dbbind(dbproc, 2, NTBSTRINGBIND, sizeof(pOrderStatus-
>c_last),
                     pOrderStatus->c_last);
                dbbind(dbproc, 3, NTBSTRINGBIND, sizeof(pOrderStatus-
>c_first),
                     pOrderStatus->c_first);
                dbbind(dbproc, 4, NTBSTRINGBIND, sizeof(pOrderStatus-
>c_middle),
                     pOrderStatus->c_middle);
                dbbind(dbproc, 5, FLT8BIND,  0, (BYTE *) &pOrderStatus-
>c_balance);
                dbbind(dbproc, 6, INTBIND,   0, (BYTE *) &pOrderStatus-
>o_id);
#ifdef NEW_SP
                dbbind(dbproc, 7, DATETIMEBIND, 0, (BYTE *)
&tmp_datetime);
#else
                dbbind(dbproc, 7, NTBSTRINGBIND, sizeof(tmpbuf),
tmpbuf);
#endif
                dbbind(dbproc, 8, INTBIND,   0, (BYTE *) &pOrderStatus-
>o_carrier_id);

        if (dbnextrow(dbproc) == REG_ROW) {
                  dbcanquery(dbproc);
#ifdef NEW_SP
                  dbdatecrack( NULL, &tmp_daterec, &tmp_datetime);
                  COPYDATE(pOrderStatus->o_entry_d, tmp_daterec);
#else
                  convert_date_and_time(&pOrderStatus->o_entry_d, tmpbuf);
#endif
                }
      }
    }
    else {
      return ERR_DB_ERROR;
    }
    if (SQLDetectDeadlock(dbproc)) {
      Sleep(DEADLOCKWAIT*tryit);
    }
    else {
      if( pOrderStatus->o_ol_cnt == 0 ||
                ( pOrderStatus->c_id == INVALID_C_ID &&
```

```
                pOrderStatus->c_last[0] == '\0' )){
              Status = ERR_DB_NOT_COMMITED;
              break;
    }
    else
              if( DBNOERR != pUD->iDBStatus ) {
                Status = ERR_DB_ERROR;
                break;
              }
              else {
                Status = ERR_DB_SUCCESS;
                break;
              }
  }
 }
 TRANSACTION_DEBUG_STAGE( pOrderStatus, LEAVING_DB );


 return( Status );
}



BOOL
SQLDetectDeadlock(DBContext dbproc)
{
 UserData   *pUD;

 if ( (pUD = (UserData *)dbgetuserdata(dbproc)) ) {
  if ( pUD->bDeadlock ) {
    pUD->bDeadlock = FALSE;
    return TRUE;
  }
 }
 return FALSE;
}

int
TPCCGetLastDBErrorDB( DBContext dbproc )
{
 UserData   *pUD;

 if ( (pUD = (UserData *)dbgetuserdata(dbproc)) ) {
  return ( pUD->iDBStatus );
 }
 return DBNOERR;
}

int
TPCCDeliveryDB( DBContext dbproc, pDeliveryData pDelivery )
{
 int                    i,rc;
 int                    deadlock_count;
 int                    always_one = 1;

 TRANSACTION_DEBUG_STAGE( pDelivery, IN_DB );

 deadlock_count = 0;


 while ( TRUE ){
  if (dbrpcinit(dbproc, "delivery", 0) == SUCCEED) {
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1, (BYTE *) &pDelivery-
>w_id);
    dbrpcparam(dbproc, NULL, 0, SYBINT2, -1, -1,
                        (BYTE *) &pDelivery->o_carrier_id);

            if (dbrpcsend(dbproc) == SUCCEED && dbsqlok(dbproc) ==
SUCCEED ){
              for (i = 0; i < 10; i++) {
                if (dbresults(dbproc) == SUCCEED) {
                  dbbind(dbproc, 1, INTBIND, 0, (BYTE *) &pDelivery-
>o_id[i]);
                  if (rc=dbnextrow(dbproc) == REG_ROW)
                        dbcanquery(dbproc);
                }

              }
            }
            else {
                  return ERR_DB_ERROR;
                }
  }
  if ( !SQLDetectDeadlock(dbproc) )
    break;
  deadlock_count++;
  Sleep(10 * deadlock_count);
 }
 TRANSACTION_DEBUG_STAGE( pDelivery, LEAVING_DB );

 return ERR_DB_SUCCESS;
}



int
TPCCCheckpointDB( DBContext dbproc, pCheckpointData pCheckpoint )
{
 UserData *pUD;

 if ( (pUD = (UserData *)dbgetuserdata(dbproc)) ) {
  pUD->iDBStatus = DBNOERR;
 }

#if 0
 if( SUCCEED == dbcmd( dbproc, "cp" ) &&
     SUCCEED == dbsqlexec( dbproc )) {
  while( NO_MORE_RESULTS != dbresults( dbproc ))
    while( NO_MORE_ROWS != dbnextrow( dbproc )) ;
 }
 else return ERR_DB_ERROR;

#else
 if( SUCCEED == dbcmd( dbproc, "dbcc tune(maxwritedes,50)" ) &&
     SUCCEED == dbsqlexec( dbproc )) {
  while( NO_MORE_RESULTS != dbresults( dbproc ))
    while( NO_MORE_ROWS != dbnextrow( dbproc )) ;
 }
 else return ERR_DB_ERROR;

 if( SUCCEED == dbcmd( dbproc, "checkpoint" ) &&
     SUCCEED == dbsqlexec( dbproc )) {
  while( NO_MORE_RESULTS != dbresults( dbproc ))
    while( NO_MORE_ROWS != dbnextrow( dbproc )) ;
 }
 else return ERR_DB_ERROR;

 if( SUCCEED == dbcmd( dbproc, "dbcc tune(maxwritedes,10)" ) &&
     SUCCEED == dbsqlexec( dbproc )) {
  while( NO_MORE_RESULTS != dbresults( dbproc ))
    while( NO_MORE_ROWS != dbnextrow( dbproc )) ;
 }
 else return ERR_DB_ERROR;
#endif

 return ERR_DB_SUCCESS;
}


void
convert_date_and_time ( DBDateData *dstruct, char *dstring)
{
 dstruct->day = UtilConvert2Char (0, dstring);
 dstruct->month = UtilConvert2Char (3, dstring);
 dstruct->year = UtilConvert2Char (6, dstring)*100 +
UtilConvert2Char(8,dstring);

 dstruct->hour = UtilConvert2Char (11, dstring);
 dstruct->minute = UtilConvert2Char (14, dstring);
 dstruct->second = UtilConvert2Char (17, dstring);
}

void
convert_date ( DBDateData *dstruct, char *dstring)
{
 dstruct->day = UtilConvert2Char (0, dstring);
 dstruct->month = UtilConvert2Char (3, dstring);
 dstruct->year = UtilConvert2Char (6, dstring)*100 +
UtilConvert2Char(8,dstring);
}
```

tm_util.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
 *                                                          *
 *                                                          *

**********************************************************
*************-*/

/*+
 * Abstract: This module contains the functions that are called by the UI
 *               server modules.
 *
 * Author: Bill Carr
 * Creation Date: Feb 1999
 *
 * Modified history:
 *
 *
 */

#ifdef _WIN32
# include <windows.h>
#else
# ifdef dunix
#  include <sys/syslimits.h>
# endif
#endif

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

#include <transpool.h>
#include <config.h>
#include <dynlib.h>

#define TM_UTIL_C
#include <tm_util.h>

static HINSTANCE        TMhInstance = NULL;

int
TPCCTMLoad( void )
{
#ifdef TM_UTIL_STATIC
  pTPCCCloseLog             = TPCCCloseLog;
  pDeleteTransactionPool    = DeleteTransactionPool;
  pTPCCShutdown             = TPCCShutdown;
  pTPCCLog                  = TPCCLog;
  pTPCCStartup              = TPCCStartup;
  pMakeTransactionPool      = MakeTransactionPool;
  pTPCCGetTransportMode     = TPCCGetTransportMode;
  pTPCCOpenLog              = TPCCOpenLog;
  pTPCCErr                  = TPCCErr;
  pTPCCErrString            = TPCCErrString;
  pUnreserveTransactionStruct = UnreserveTransactionStruct;
  pTPCCDelivery             = TPCCDelivery;
  pReserveTransactionStruct = ReserveTransactionStruct;
  pTPCCNewOrder             = TPCCNewOrder;
  pTPCCOrderStatus          = TPCCOrderStatus;
  pTPCCPayment              = TPCCPayment;
  pTPCCStockLevel           = TPCCStockLevel;
  pTPCCCheckpoint           = TPCCCheckpoint;
  pTPCCCheckpointDisconnect = TPCCCheckpointDisconnect;
  pTPCCCheckpointConnect    = TPCCCheckpointConnect;
  pTPCCDisconnect           = TPCCDisconnect;
  pTPCCConnect              = TPCCConnect;

  return( ERR_SUCCESS );
#else
  int                               Status = ERR_SUCCESS;
  char                              TMModuleName[PATH_MAX];
  int                               size;
  char                              *errstr;

  size = sizeof( TMModuleName ) - 1;
  if( ERR_SUCCESS != GetConfigStr( FFE_TM_MODULE, &size,
TMModuleName ))
  {
    Status = ERR_CANT_FIND_TM_VALUE;
  }
#pragma message ("FIXME: There appears to be a problem finding the TM
module by just using its name without a path.  How can we get it to load
under all different callers (not just inetinfo)?")
  else if( NULL == ( TMhInstance = DynamicLibraryLoad(
TMModuleName ))
  {
    errstr = DynamicLibraryError( );
    Status = ERR_CANT_LOAD_TM_MODULE;
  }
  else if( NULL == ( pMakeTransactionPool = (MakeTransactionPoolFunc)
                                DynamicLibrarySymbol( TMhInstance,
"MakeTransactionPool" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pReserveTransactionStruct =
(ReserveTransactionStructFunc)
                                DynamicLibrarySymbol( TMhInstance,
"ReserveTransactionStruct" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pUnreserveTransactionStruct =
(UnreserveTransactionStructFunc)
                                DynamicLibrarySymbol( TMhInstance,
"UnreserveTransactionStruct" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pDeleteTransactionPool =
(DeleteTransactionPoolFunc)
                                DynamicLibrarySymbol( TMhInstance,
"DeleteTransactionPool" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCStartup = (TPCCStartupFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCStartup" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCConnect = (TPCCConnectFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCConnect" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
```

```
  else if( NULL == ( pTPCCCheckpointConnect =
(TPCCCheckpointConnectFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCCheckpointConnect" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCDelivery = (TPCCDeliveryFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCDelivery" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCNewOrder = (TPCCNewOrderFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCNewOrder" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCOrderStatus = (TPCCOrderStatusFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCOrderStatus" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCPayment = (TPCCPaymentFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCPayment" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCStockLevel = (TPCCStockLevelFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCStockLevel" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCCheckpoint = (TPCCCheckpointFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCCheckpoint" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCCheckpointDisconnect =
(TPCCCheckpointDisconnectFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCCheckpointDisconnect" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCDisconnect = (TPCCDisconnectFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCDisconnect" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCShutdown = (TPCCShutdownFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCShutdown" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCOpenLog = (TPCCOpenLogFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCOpenLog" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCErr = (TPCCErrFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCErr" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCErrString = (TPCCErrStringFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCErrString" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCLog = (TPCCLogFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCLog" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCCloseLog = (TPCCCloseLogFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCCloseLog" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }
  else if( NULL == ( pTPCCGetTransportMode =
(TPCCGetTransportModeFunc)
                                DynamicLibrarySymbol( TMhInstance,
"TPCCGetTransportMode" )))
  {
    Status = ERR_CANT_FIND_TM_FUNC;
  }

  return( Status );
#endif
}


TPCCTMUnload( void )
{
#ifdef TM_UTIL_STATIC
  return( ERR_SUCCESS );
#else
  int                                   Status = ERR_SUCCESS;

#ifdef _WIN32
  if( !DynamicLibraryUnload( TMhInstance ))
  {
    Status = ERR_CANT_UNLOAD_TM_MODULE;
  }
#else
#pragma mesasge ("FIXME: Unload crashes on UNIX.")
#endif

  return( Status );
#endif
}
```

tm_util.h

```
#ifndef TM_UTIL_H
#define TM_UTIL_H
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.               *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.                *
```

```
 *                                                                  *
 *                                                                  *

********************************************************************
*************-*/

/*+
 * Abstract: This module contains the functions that are called by the UI
 *               server modules.
 *
 * Author: Bill Carr
 * Creation Date: Feb 1999
 *
 * Modified history:
 *
 *
 */


#ifdef TM_UTIL_C
#define TMGLOBAL(thing,initializer) thing = initializer
#else
#define TMGLOBAL(thing,initializer) extern thing
#endif

TMGLOBAL(TPCCStartupFunc pTPCCStartup, NULL);
TMGLOBAL(TPCCConnectFunc pTPCCConnect, NULL);
TMGLOBAL(TPCCDeliveryFunc pTPCCDelivery, NULL);
TMGLOBAL(TPCCNewOrderFunc pTPCCNewOrder, NULL);
TMGLOBAL(TPCCOrderStatusFunc pTPCCOrderStatus, NULL);
TMGLOBAL(TPCCPaymentFunc pTPCCPayment, NULL);
TMGLOBAL(TPCCStockLevelFunc pTPCCStockLevel, NULL);
TMGLOBAL(TPCCDisconnectFunc pTPCCDisconnect, NULL);
TMGLOBAL(TPCCShutdownFunc pTPCCShutdown, NULL);
TMGLOBAL(TPCCCheckpointConnectFunc pTPCCCheckpointConnect,
NULL);
TMGLOBAL(TPCCCheckpointFunc pTPCCCheckpoint, NULL);
TMGLOBAL(TPCCCheckpointDisconnectFunc
pTPCCCheckpointDisconnect, NULL);
TMGLOBAL(TPCCErrStringFunc pTPCCErrString, NULL);
TMGLOBAL(TPCCOpenLogFunc pTPCCOpenLog, NULL);
TMGLOBAL(TPCCErrFunc pTPCCErr, NULL);
TMGLOBAL(TPCCLogFunc pTPCCLog, NULL);
TMGLOBAL(TPCCCloseLogFunc pTPCCCloseLog, NULL);
TMGLOBAL(TPCCGetTransportModeFunc pTPCCGetTransportMode,
NULL);
TMGLOBAL(DeleteTransactionPoolFunc pDeleteTransactionPool, NULL);
TMGLOBAL(MakeTransactionPoolFunc pMakeTransactionPool, NULL);
TMGLOBAL(ReserveTransactionStructFunc pReserveTransactionStruct,
NULL);
TMGLOBAL(UnreserveTransactionStructFunc
pUnreserveTransactionStruct, NULL);



#endif
```

transpool.c

```
/*+*****************************************************************
***************
 *                                                                  *
 * COPYRIGHT (c) 1997, 2000 BY                                      *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 * 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
DIGITAL EQUIPMENT   *
 * CORPORATION.                                                     *
 *                                                                  *
 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS   *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.             *
 *                                                                  *
 *                                                                  *

********************************************************************
*************-*/

/*+
 * Abstract: This is the source file for transaction pool.
 *
 * Author: WCarr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
-*/

#define TRANSPOOL_C

#ifdef _WIN32
# include <windows.h>
#else
# define BOOL int
#endif

#include <stdlib.h>
#ifdef TRANSACTION_DEBUG
# include <crtdbg.h>
#endif

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

#include <transpool.h>

#ifdef TRANSACTION_DEBUG_HISTORY

# define HISTORY_SIZE ((int)( 5000 *   1.2 *   0.1 *  60 * 2.22222 *      4
))
#endif


#ifdef USE_PROCESSES
  static int                    NextDeliveryStruct = 0;
#else
typedef struct _TransactionPoolStruct
{
  CRITICAL_SECTION critSec;
  int           iNextFree;
#ifdef TRANSACTION_DEBUG
  int           iMaxIndex;
  int           iTransactionSize;
#ifdef TRANSACTION_DEBUG_HISTORY
  int           iHistoryId;
  struct
  {
    int                         iOpCode;
    int                         iFailure;
    int                         iReserveHistoryId;
    int                         iUnreserveHistoryId;
    int                         iType;
#ifdef TRANSACTION_DEBUG_THREAD_CHECK
    DWORD dwThreadId;
    DWORD dwTMThreadId;
#endif
    void        *pTrans;
  }                             History[HISTORY_SIZE];
#endif
#endif
```

```
  void          *index[1];
  char          data[1];
} TransactionPoolStruct;


static pTransactionPoolStruct              gpTransactionPool = { 0 };

#endif




void
MakeTransactionPool( int dwTransactionPoolSize )
{
#ifndef USE_PROCESSES
  int iMaxSize;
  int iSize;
  char *data;
  int ii;
  pTransactionStruct        pTrans = NULL;


  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));
#ifdef TRANSACTION_DEBUG
#ifdef TRANSACTION_DEBUG_HISTORY
  iMaxSize += (char *)&pTrans->Generic.info.conn - (char *)pTrans;
#endif
#endif
  iSize = (((char *)&gpTransactionPool->index - (char *)gpTransactionPool)
+
                  (((char *)gpTransactionPool->data - (char
*)gpTransactionPool->index)
                   * dwTransactionPoolSize ) +
                  (sizeof( char ) * iMaxSize * dwTransactionPoolSize ));
  gpTransactionPool = malloc( iSize );
  InitializeCriticalSection( &gpTransactionPool->critSec );
#ifdef TRANSACTION_DEBUG
  gpTransactionPool->iMaxIndex = dwTransactionPoolSize - 1;
  gpTransactionPool->iTransactionSize = iMaxSize;
#ifdef TRANSACTION_DEBUG_HISTORY
  gpTransactionPool->iHistoryId = -1;
#endif
#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;
  }
#endif
}


void
DeleteTransactionPool( void )
{
#ifndef USE_PROCESSES
  DeleteCriticalSection( &gpTransactionPool->critSec );
  free( gpTransactionPool );
#endif
}

void *
ReserveTransactionStruct( int type )
{
#ifdef USE_PROCESSES
  static TransactionStruct Trans;
#else
# define NextDeliveryStruct 0
#endif

  pTransactionStruct        pTrans;
  void                      *pStruct;
#ifdef TRANSACTION_DEBUG_HISTORY
  int                       iHistoryId;
#endif

#ifdef USE_PROCESSES
  pTrans = &Trans;
#else
  EnterCriticalSection( &gpTransactionPool->critSec );
  pTrans = gpTransactionPool->index[gpTransactionPool->iNextFree];

#ifdef TRANSACTION_DEBUG
  _ASSERT( gpTransactionPool->iNextFree <= gpTransactionPool-
>iMaxIndex );
  memset( pTrans, 0x01, gpTransactionPool->iTransactionSize );
#endif
#endif

  switch( type )
  {
  case DELIVERY_TRANS:
    pTrans->Generic.info.delivery[NextDeliveryStruct].iType = type;
    pTrans->Generic.info.delivery[NextDeliveryStruct].iSize =
      sizeof( DeliveryData );
    pStruct = &pTrans->Generic.info.delivery[NextDeliveryStruct];
    break;
  case NEW_ORDER_TRANS:
    pTrans->Generic.info.newOrder.iType = type;
    pTrans->Generic.info.newOrder.iSize = sizeof( NewOrderData );
    pStruct = &pTrans->Generic.info.newOrder;
    break;
  case ORDER_STATUS_TRANS:
    pTrans->Generic.info.orderStatus.iType = type;
    pTrans->Generic.info.orderStatus.iSize = sizeof( OrderStatusData );
    pStruct = &pTrans->Generic.info.orderStatus;
    break;
  case PAYMENT_TRANS:
    pTrans->Generic.info.payment.iType = type;
    pTrans->Generic.info.payment.iSize = sizeof( PaymentData );
    pStruct = &pTrans->Generic.info.payment;
    break;
  case STOCK_LEVEL_TRANS:
    pTrans->Generic.info.stockLevel.iType = type;
    pTrans->Generic.info.stockLevel.iSize = sizeof( StockLevelData );
    pStruct = &pTrans->Generic.info.stockLevel;
    break;
  case GENERIC_TRANS:
    pTrans->Generic.info.conn.iType = type;
    pTrans->Generic.info.conn.iSize = sizeof( GenericData );
    pStruct = &pTrans->Generic.info.conn;
    break;
  case CONNECT_TRANS:
    pTrans->Generic.info.login.iType = type;
    pTrans->Generic.info.login.iSize = sizeof( LoginData );
    pStruct = &pTrans->Generic.info.login;
    break;
  case DISCONNECT_TRANS:
    pTrans->Generic.info.conn.iType = type;
    pTrans->Generic.info.conn.iSize = sizeof( ConnData );
    pStruct = &pTrans->Generic.info.conn;
    break;
  case CHECKPOINT_CONNECT_TRANS:
    pTrans->Generic.info.login.iType = type;
    pTrans->Generic.info.login.iSize = sizeof( LoginData );
    pStruct = &pTrans->Generic.info.login;
    break;
  case CHECKPOINT_TRANS:
    pTrans->Generic.info.checkpoint.iType = type;
    pTrans->Generic.info.checkpoint.iSize = sizeof( CheckpointData );
    pStruct = &pTrans->Generic.info.checkpoint;
    break;
```

```
  case CHECKPOINT_DISCONNECT_TRANS:
    pTrans->Generic.info.conn.iType = type;
    pTrans->Generic.info.conn.iSize = sizeof( ConnData );
    pStruct = &pTrans->Generic.info.conn;
    break;
  default:
    pTrans->Generic.info.conn.iType = type;
    pTrans->Generic.info.conn.iSize = 0;
    pStruct = &pTrans->Generic.info.conn;
    break;
  }

#ifdef USE_PROCESSES
  if( DELIVERY_TRANS == type )
  {

    NextDeliveryStruct++;
  }
#else
#ifdef TRANSACTION_DEBUG
#ifdef TRANSACTION_DEBUG_HISTORY
  gpTransactionPool->iHistoryId++;
  iHistoryId = gpTransactionPool->iHistoryId % HISTORY_SIZE;
  gpTransactionPool->History[iHistoryId].iFailure = 0;
  pTrans->Generic.info.conn.iStage = 0;
#ifdef TRANSACTION_DEBUG_THREAD_CHECK
  pTrans->dwThreadId = GetCurrentThreadId();
  pTrans->dwTMThreadId = 0;
  pTrans->iSynchronous = 1;
#endif
  pTrans->iReserveHistoryId = iHistoryId;
  pTrans->iUnreserveHistoryId = 0;
  gpTransactionPool->History[iHistoryId].iOpCode = 1;
  gpTransactionPool->History[iHistoryId].iReserveHistoryId = iHistoryId;
  gpTransactionPool->History[iHistoryId].iUnreserveHistoryId = 0;
  gpTransactionPool->History[iHistoryId].iType = type;
#ifdef TRANSACTION_DEBUG_THREAD_CHECK
  gpTransactionPool->History[iHistoryId].dwThreadId = pTrans-
>dwThreadId;
  gpTransactionPool->History[iHistoryId].dwTMThreadId = pTrans-
>dwTMThreadId;
#endif
  gpTransactionPool->History[iHistoryId].pTrans = pTrans;
#endif
#endif

  gpTransactionPool->iNextFree++;
  LeaveCriticalSection( &gpTransactionPool->critSec );
#endif

  return( pStruct );
}


void
UnreserveTransactionStruct( int type, void **ppData )
{
#ifdef USE_PROCESSES
  if( DELIVERY_TRANS == type )
  {

    NextDeliveryStruct--;
  }
#else
  pTransactionStruct        pTrans = NULL;
#ifdef TRANSACTION_DEBUG_HISTORY
  int                                   iHistoryId;
#endif

  pTrans = (pTransactionStruct)((char *)(*ppData) -
                                        ((char *)&pTrans->Generic.info.conn
- (char *)pTrans));

  EnterCriticalSection( &gpTransactionPool->critSec );

#ifdef TRANSACTION_DEBUG
  _ASSERT( gpTransactionPool->iNextFree > 0 );
#ifdef TRANSACTION_DEBUG_HISTORY
  gpTransactionPool->iHistoryId++;
  iHistoryId = gpTransactionPool->iHistoryId % HISTORY_SIZE;
  gpTransactionPool->History[iHistoryId].iFailure++;
  gpTransactionPool->History[iHistoryId].iFailure++;

#ifdef TRANSACTION_DEBUG_THREAD_CHECK
  if( DELIVERY_TRANS != type )
  {
    if( pTrans->iSynchronous == 1 )
      _ASSERT((pTrans->dwThreadId == GetCurrentThreadId( )));
    else if( pTrans->iSynchronous == 0 )
      _ASSERT((pTrans->dwTMThreadId == GetCurrentThreadId( )));
    else
      _ASSERT(FALSE);
  }
#endif
  gpTransactionPool->History[iHistoryId].iFailure++;
  _ASSERT((pTrans->Generic.info.conn.iType==type));
  switch( type )
  {
  case DELIVERY_TRANS:
    _ASSERT( sizeof( DeliveryData ) == pTrans->Generic.info.conn.iSize );
    break;
  case NEW_ORDER_TRANS:
    _ASSERT( sizeof( NewOrderData ) == pTrans->Generic.info.conn.iSize
);
    break;
  case ORDER_STATUS_TRANS:
    _ASSERT( sizeof( OrderStatusData ) == pTrans->Generic.info.conn.iSize
);
    break;
  case PAYMENT_TRANS:
    _ASSERT( sizeof( PaymentData ) == pTrans->Generic.info.conn.iSize );
    break;
  case STOCK_LEVEL_TRANS:
    _ASSERT( sizeof( StockLevelData ) == pTrans->Generic.info.conn.iSize
);
    break;
  case GENERIC_TRANS:
    _ASSERT( sizeof( GenericData ) == pTrans->Generic.info.conn.iSize );
    break;
  case CONNECT_TRANS:
    _ASSERT( sizeof( LoginData ) == pTrans->Generic.info.conn.iSize );
    break;
  case DISCONNECT_TRANS:
    _ASSERT( sizeof( ConnData ) == pTrans->Generic.info.conn.iSize );
    break;
  case CHECKPOINT_CONNECT_TRANS:
    _ASSERT( sizeof( LoginData ) == pTrans->Generic.info.conn.iSize );
    break;
  case CHECKPOINT_TRANS:
    _ASSERT( sizeof( CheckpointData ) == pTrans->Generic.info.conn.iSize
);
    break;
  case CHECKPOINT_DISCONNECT_TRANS:
    _ASSERT( sizeof( ConnData ) == pTrans->Generic.info.conn.iSize );
    break;
  default:
    _ASSERT( 0 == pTrans->Generic.info.conn.iSize );
    break;
  }
  gpTransactionPool->History[iHistoryId].iFailure++;
  _ASSERT((gpTransactionPool->History[pTrans-
>iReserveHistoryId].pTrans) == pTrans);
  pTrans->iUnreserveHistoryId = iHistoryId;
  gpTransactionPool->History[iHistoryId].iOpCode = 2;
  gpTransactionPool->History[iHistoryId].iReserveHistoryId = pTrans-
>iReserveHistoryId;
  gpTransactionPool->History[iHistoryId].iUnreserveHistoryId = iHistoryId;
  gpTransactionPool->History[iHistoryId].iType = type;
#ifdef TRANSACTION_DEBUG_THREAD_CHECK
  gpTransactionPool->History[iHistoryId].dwThreadId = pTrans-
>dwThreadId;
  gpTransactionPool->History[iHistoryId].dwTMThreadId = pTrans-
>dwTMThreadId;
#endif
  gpTransactionPool->History[iHistoryId].pTrans = pTrans;
#endif
#endif

  gpTransactionPool->index[--gpTransactionPool->iNextFree] = pTrans;
  LeaveCriticalSection( &gpTransactionPool->critSec );
```

```
#endif
  *ppData = NULL;
}
```

transpool.h

```
#ifndef TRANSPOOL_H
#define TRANSPOOL_H

/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
*************-*/

/*+
 * Abstract: This is the header file for transpool.c.
 *
 * Author: WCarr
 * Creation Date: May 1998
 *
 * Modified history:
 *
 *
 */

#ifdef _WIN32
typedef struct _TransactionPoolStruct *pTransactionPoolStruct;
#endif


#pragma message ("FIXME: Transpool.h has its own (inconsistant)
transaction defintion numbers.  However an effort is underway to
standardize on the definitions in this file because the need to define a
differing trans set for every workload should be localized for replacement so
the test harness can be minimally (hopefully totally) unaffected.")

#define DELIVERY_TRANS 0
#define NEW_ORDER_TRANS 1
#define ORDER_STATUS_TRANS 2
#define PAYMENT_TRANS 3
#define STOCK_LEVEL_TRANS 4
#define GENERIC_TRANS 5
#define CONNECT_TRANS 6
#define DISCONNECT_TRANS 7
#define CHECKPOINT_CONNECT_TRANS 8
#define CHECKPOINT_TRANS 9
#define CHECKPOINT_DISCONNECT_TRANS 10
#define HIGHEST_TRANS 11

#ifdef USE_PROCESSES
typedef struct _TransactionStruct
{
  GenericData               Generic;
} TransactionStruct, *pTransactionStruct;
# define TRANSACTION_DEBUG_TM_THREAD_SET(pTxn)
# define TRANSACTION_DEBUG_TM_SYNC_SET(pTxn,sync)
#else
typedef struct _TransactionStruct
{
#ifdef TRANSACTION_DEBUG
#ifdef TRANSACTION_DEBUG_HISTORY
  int                       iReserveHistoryId;
  int                       iUnreserveHistoryId;
#ifdef TRANSACTION_DEBUG_THREAD_CHECK
  DWORD                     dwThreadId;
  DWORD                     dwTMThreadId;
  int                       iSynchronous;
#endif
#endif
#endif
  GenericData               Generic;
} TransactionStruct, *pTransactionStruct;

#if defined TRANSACTION_DEBUG && \
    defined TRANSACTION_DEBUG_HISTORY && \
    defined TRANSACTION_DEBUG_THREAD_CHECK
# define TRANSACTION_DEBUG_TM_THREAD_SET(pTxn) \
              ((pTransactionStruct) \
               ((char *)(pTxn) - \
                ((char *)(&((pTransactionStruct)NULL)->Generic) - \
                 (char *)(pTransactionStruct)NULL)))->dwTMThreadId \
                          = GetCurrentThreadId( )
# define TRANSACTION_DEBUG_TM_SYNC_SET(pTxn,sync) \
              ((pTransactionStruct) \
               ((char *)(pTxn) - \
                ((char *)(&((pTransactionStruct)NULL)->Generic) - \
                 (char *)(pTransactionStruct)NULL)))->iSynchronous \
                          = (sync)
#else
# define TRANSACTION_DEBUG_TM_THREAD_SET(pTxn)
# define TRANSACTION_DEBUG_TM_SYNC_SET(pTxn,sync)
#endif



#endif


void MakeTransactionPoolUI( int dwTransactionPoolSize );
typedef void (*MakeTransactionPoolUIFunc)( int dwTransactionPoolSize );
void MakeTransactionPool( int dwTransactionPoolSize );
typedef void (*MakeTransactionPoolFunc)( int dwTransactionPoolSize );

void DeleteTransactionPoolUI( void );
typedef void (*DeleteTransactionPoolUIFunc)( void );
void DeleteTransactionPool( void );
typedef void (*DeleteTransactionPoolFunc)( void );

void *ReserveTransactionStructUI( int type );
typedef void *(*ReserveTransactionStructUIFunc)( int type );
void *ReserveTransactionStruct( int type );
typedef void *(*ReserveTransactionStructFunc)( int type );

void UnreserveTransactionStructUI( int type, void **pData );
typedef void (*UnreserveTransactionStructUIFunc)( int type, void **pData
);
void UnreserveTransactionStruct( int type, void **pData );
typedef void (*UnreserveTransactionStructFunc)( int type, void **pData );

#endif
```

tux_cli.c

```
/*+**********************************************************
**************
```

```
 *                                                                *
 *  COPYRIGHT (c) 1997, 2000 BY                                   *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                 *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                                  *
 *                                                                *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.               *
 *                                                                *
 *                                                                *

******************************************************************
*************-*/



#ifdef _WIN32
# include <windows.h>
#else
# define BOOL int
# include <sys/time.h>
#endif

#include <stdlib.h>

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

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


#include <atmi.h>


static DWORD tlsIndex;
static CRITICAL_SECTION TLS_crit_sec;


int
IsTuxInit()
{
 TPINIT *tpinitbuf;

 int x = 1;
 int retcode = -1;
 int count = 0;
 static int num_tpinits = 0;

 if(!(TlsGetValue(tlsIndex)))
  {
   EnterCriticalSection(&TLS_crit_sec);
   while(count < 20)
   {
                if(NULL == (tpinitbuf = (TPINIT *) tpalloc("TPINIT",
NULL,

sizeof(TPINIT))))
                {
                 TPCCErr("error with tpalloc - %d - %d", tperrno,count);
                }
                else
                {
                 tpinitbuf->flags |= TPMULTICONTEXTS;
                 itoa(++num_tpinits, tpinitbuf->cltname, 10);
                 retcode = tpinit(tpinitbuf);
                 if(-1 != retcode)
                 {
                  TlsSetValue(tlsIndex, &x);
                  tpfree((char*)tpinitbuf);
                  break;
                 }
                 else
                 {
                  TPCCErr("error with TPINIT - %s (%d) - %d\n\t\t..%s..",
                                tpstrerror(tperrno),
                                tperrno,
                                count,
                                tpstrerrordetail( tperrordetail( 0 ), 0 ));
                  tpfree((char*)tpinitbuf);
                 }
                }

                count++;
                if(count > 50)
                 {
                  retcode = -1;
                  TPCCErr("exceeded 50 trys in TPINIT");
                 }

                Sleep(50);
    }
    LeaveCriticalSection(&TLS_crit_sec);
    Sleep(50);
    if( -1 != retcode)
                return ERR_DB_SUCCESS;
    else
                return(retcode);
  }
 return ERR_DB_SUCCESS;
}



int
TPCCGetTransportMode( int *Mode )
{
 *Mode = TRANS_SYNC;

 return( ERR_SUCCESS );
}


int
TPCCStartup( pStartupData pStartup )
{
 int status;

 InitializeCriticalSection (&TLS_crit_sec);


 status = system("tmboot -y");
 if (status != 0)
  {
   TPCCErr("Error booting the tuxedo servers.");
   return status;
  }


 status = DELIClientStartup( &pStartup->DeliveryTransport,
                                &pStartup->Login, pStartup-
>loginDelay,
                                pStartup->Path );
 if( ERR_SUCCESS != status )
  return status;


 tlsIndex = TlsAlloc();
```

```
  if(0 > tlsIndex)
    {
      TPCCErr("TPCCStartup error - thread local storage index alloc failed");
      return ERR_CANT_ALLOCATE_THREAD_LOCAL_STORAGE;
    }
  return ERR_SUCCESS;
}



int
TPCCConnect( pLoginData *ppLogin )
{
  return ERR_DB_SUCCESS;
}


int
TuxShutdown()
{
  return ERR_DB_SUCCESS;
}


int
TPCCShutdown( void )
{
  int retcode;

  if(0 == (TlsFree(tlsIndex))) TPCCErr("Error freeing TLS for tpinit");


  retcode = system("tmshutdown -y");
  if (retcode != 0)
    {
      TPCCErr("Error shutting the tuxedo servers down.");
      return retcode;
    }

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

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

  return(TuxShutdown());
}


int
TPCCDisconnect( pConnData *ppConn )
{
  return ERR_DB_SUCCESS;
}



int
TPCCDeliveryDeferred( pDeliveryData *ppDelivery )
{
  int retcode = ERR_DB_SUCCESS;

  pDeliveryData retptr;
  int dysiz = sizeof(DeliveryData);


  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
      TPCCErr("IsTuxInit - delivery ");
      return ERR_DB_ERROR;
    }


  if(NULL == ( retptr= (pDeliveryData) tpalloc("CARRAY", NULL,
dysiz)))
    {
      TPCCErr("tp alloc in delivery");
      return ERR_DB_ERROR;
    }
  memcpy( retptr, *ppDelivery, dysiz);


  retcode = tpcall("dy_transaction", (char *)retptr, dysiz,
                                (char**)&retptr, &dysiz, TPSIGRSTRT);
  if( -1 == retcode )
    {
      TPCCErr("tpcall - delivery: %d", tperrno);
      tpfree((char*) retptr);
      return ERR_DB_ERROR;
    }
  memcpy(*ppDelivery, retptr, dysiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}


int
TPCCNewOrder( pNewOrderData *ppNewOrder )
{
  int retcode = ERR_DB_SUCCESS;

  pNewOrderData retptr;
  int nosiz = sizeof(NewOrderData);


  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
      TPCCErr("IsTuxInit - new order: %d ", tperrno);
      return ERR_DB_ERROR;
    }


  if(NULL == ( retptr= (pNewOrderData) tpalloc("CARRAY", NULL,
nosiz)))
    {
      TPCCErr("tp alloc in neworder: %d ", tperrno);
      return ERR_DB_ERROR;
    }
  memcpy( retptr, *ppNewOrder, nosiz);


  retcode = tpcall("no_transaction", (char *)retptr, nosiz,
                                (char**)&retptr, &nosiz, TPSIGRSTRT);
  if( -1 == retcode )
    {
      TPCCErr("tpcall - new order: %d", tperrno);
      tpfree((char*) retptr);
      return ERR_DB_ERROR;
    }
  memcpy(*ppNewOrder, retptr, nosiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}


int
TPCCOrderStatus( pOrderStatusData *ppOrderStatus )
{
  int retcode = ERR_DB_SUCCESS;

  pOrderStatusData retptr;
  int ossiz = sizeof(OrderStatusData);


  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
      TPCCErr("IsTuxInit - order status");
      return ERR_DB_ERROR;
    }


  if(NULL == ( retptr= (pOrderStatusData) tpalloc("CARRAY", NULL,
ossiz)))
    {
      TPCCErr("tp alloc in order status: %d", tperrno);
      return ERR_DB_ERROR;
    }
  memcpy( retptr, *ppOrderStatus, ossiz);
```

```
  retcode = tpcall("os_transaction", (char *)retptr, ossiz,
                                (char**)&retptr, &ossiz, TPSIGRSTRT);
  if( -1 == retcode )
    {
     TPCCErr("tpcall - order status");
     tpfree((char*) retptr);
     return ERR_DB_ERROR;
    }
  memcpy(*ppOrderStatus, retptr, ossiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}


int
TPCCPayment( pPaymentData *ppPayment )
{
  int retcode = ERR_DB_SUCCESS;

  pPaymentData retptr;
  long ptsiz = sizeof(PaymentData);


  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
     TPCCErr("IsTuxInit - payment ");
     return ERR_DB_ERROR;
    }


  if(NULL == ( retptr= (pPaymentData) tpalloc("CARRAY", NULL, ptsiz)))
    {
     TPCCErr("tp alloc in payment");
     return ERR_DB_ERROR;
    }
  memcpy( retptr, *ppPayment, ptsiz);


  retcode = tpcall("pt_transaction", (char *)retptr, ptsiz,
                                (char**)&retptr, &ptsiz, TPSIGRSTRT);
  if( -1 == retcode )
    {
     TPCCErr("tpcall - payment: %d ", tperrno);
     tpfree((char*) retptr);
     return ERR_DB_ERROR;
    }
  memcpy(*ppPayment, retptr, ptsiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}


int
TPCCStockLevel( pStockLevelData *ppStockLevel )
{
  int retcode = ERR_DB_SUCCESS;

  pStockLevelData retptr;
  int slsiz = sizeof(StockLevelData);


  if( ERR_DB_SUCCESS != IsTuxInit() )
    {
     TPCCErr("IsTuxInit - stock level ");
     return ERR_DB_ERROR;
    }


  if(NULL == ( retptr= (pStockLevelData) tpalloc("CARRAY", NULL,
slsiz)))
    {
     TPCCErr("tp alloc in stock level");
     return ERR_DB_ERROR;
    }
  memcpy( retptr, *ppStockLevel, slsiz);


  retcode = tpcall("sl_transaction", (char *)retptr, slsiz,
                                (char**)&retptr, &slsiz, TPSIGRSTRT);
  if( -1 == retcode )
    {
     TPCCErr("tpcall - stock level: %d", tperrno);
     tpfree((char*) retptr);
     return ERR_DB_ERROR;
    }
  memcpy(*ppStockLevel, retptr, slsiz);
  tpfree((char*) retptr);
  return ERR_DB_SUCCESS;
}
```

tux_srv.c

```
/*+**********************************************************
***************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
*************-*/
#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <sys/types.h>
#include <windows.h>
#include <process.h>
#include <Winsock.h>

#define TUX_SRV_C

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

#include <tpcc.h>
#include <config.h>

#include <atmi.h>
#include <userlog.h>

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


DBContext DBC;
```

```
#if defined(__cplusplus)
extern "C" {
#endif
extern int _tmrunserver _((int));
extern void dy_transaction _((TPSVCINFO *));
extern void no_transaction _((TPSVCINFO *));
extern void os_transaction _((TPSVCINFO *));
extern void pt_transaction _((TPSVCINFO *));
extern void sl_transaction _((TPSVCINFO *));
#if defined(__cplusplus)
}
#endif

static struct tmdsptchtbl_t _tmdsptchtbl[] = {
        { "dy_transaction", "dy_transaction", (void (*) _((TPSVCINFO
*))) dy_transaction, 0, 0 },
        { "no_transaction", "no_transaction", (void (*) _((TPSVCINFO
*))) no_transaction, 1, 0 },
        { "os_transaction", "os_transaction", (void (*) _((TPSVCINFO
*))) os_transaction, 2, 0 },
        { "pt_transaction", "pt_transaction", (void (*) _((TPSVCINFO
*))) pt_transaction, 3, 0 },
        { "sl_transaction", "sl_transaction", (void (*) _((TPSVCINFO
*))) sl_transaction, 4, 0 },
        { NULL, NULL, NULL, 0, 0 }
};

#ifndef _TMDLLIMPORT
#define _TMDLLIMPORT
#endif

_TMDLLIMPORT extern struct xa_switch_t tmnull_switch;

struct tmsvrargs_t tmsvrargs = {
        NULL,
        &_tmdsptchtbl[0],
        0,
        tpsvrinit,
        tpsvrdone,
        _tmrunserver,
        NULL,
        NULL,
        NULL,
        NULL
};

struct tmsvrargs_t *
#ifdef _TMPROTOTYPES
_tmgetsvrargs(void)
#else
_tmgetsvrargs()
#endif
{
        tmsvrargs.xa_switch = &tmnull_switch;
        return(&tmsvrargs);
}

int
#ifdef _TMPROTOTYPES
main(int argc, char **argv)
#else
main(argc,argv)
int argc;
char **argv;
#endif
{
#ifdef TMMAINEXIT
#include "mainexit.h"
#endif

        return( _tmstartserver( argc, argv, _tmgetsvrargs()));
}
```

```
int
tpsvrinit( int argc, char *argv[] )
{
  BOOL                          bLog;
  char                          szPath[MAX_PATH];
  int                           Len = sizeof( szPath ) - 1;
  StartupData                   Startup;
  pStartupData                  pStartup = &Startup;
  int status;


  argc = argc;
  argv = argv;




  userlog("Starting tpcc server");


  pStartup->Path = szPath;
  status = TPCCGetConfig( &bLog,
                                        &Len, pStartup->Path,
                                        &pStartup->Login, &pStartup-
>loginDelay,
                                        &pStartup->Config,
                                        &pStartup->Transport,
                                        &pStartup->DeliveryTransport );
  if (ERR_SUCCESS == status)
  {

    strcpy( pStartup->Login.databaseLogin.szApplication, "TUX_SRV" );

    TPCCOpenLog( szPath );
    TPCCStartupDB( &pStartup->Config );
    pStartup->Login.status = TPCCConnectDB( &DBC, &pStartup->Login );

    if(ERR_DB_SUCCESS != pStartup->Login.status)
    {
      TPCCErr( "tpsvrinit : Error logging into db." );
      return ERR_DB_ERROR;
    }
    TPCCLog( "%s, dbprocptr = %8X\r\n",
                pStartup->Login.databaseLogin.szApplication, DBC );
  }
  else
  {
    TPCCErr("tpsvrinit : could not get configuration settings");
  }


  return (0);
}



void tpsvrdone(void)
{
  TPCCShutdownDB();
  return;
}


void
dy_transaction( TPSVCINFO *dy_wksp )
{
  pDeliveryData ptr;

  ptr = (pDeliveryData)dy_wksp->data;

  ptr->status = TPCCDeliveryDB( DBC, ptr );
  if(ERR_DB_ERROR != ptr->status)
    tpreturn(TPSUCCESS, ptr->status, dy_wksp->data, dy_wksp->len, 0);
  else
    tpreturn(TPFAIL, ptr->status, dy_wksp->data, 0L, 0);
}
```

```
void
no_transaction( TPSVCINFO *no_wksp )
{
 pNewOrderData ptr;

 ptr = (pNewOrderData)no_wksp->data;

 ptr->status = TPCCNewOrderDB( DBC, ptr );
 if(ERR_DB_ERROR != ptr->status)
   tpreturn(TPSUCCESS, ptr->status, no_wksp->data, no_wksp->len, 0);
 else
   tpreturn(TPFAIL, ptr->status, no_wksp->data, 0L, 0);
}


void
os_transaction( TPSVCINFO *os_wksp )
{
 pOrderStatusData ptr;

 ptr = (pOrderStatusData)os_wksp->data;

 ptr->status = TPCCOrderStatusDB( DBC, ptr );
 if(ERR_DB_ERROR != ptr->status)
   tpreturn(TPSUCCESS, ptr->status, os_wksp->data, os_wksp->len, 0);
 else
   tpreturn(TPFAIL, ptr->status, os_wksp->data, 0L, 0);
}


void
pt_transaction( TPSVCINFO *pt_wksp )
{
 pPaymentData ptr;

 ptr = (pPaymentData)pt_wksp->data;

 ptr->status = TPCCPaymentDB( DBC, ptr );
 if(ERR_DB_ERROR != ptr->status)
   tpreturn(TPSUCCESS, ptr->status, pt_wksp->data, sizeof(PaymentData),
0);
 else
   tpreturn(TPFAIL, ptr->status, pt_wksp->data, 0L, 0);

}


void
sl_transaction( TPSVCINFO *sl_wksp )
{
 pStockLevelData ptr;

 ptr = (pStockLevelData)sl_wksp->data;

 ptr->status = TPCCStockLevelDB( DBC, ptr );
 if(ERR_DB_ERROR != ptr->status)
   tpreturn(TPSUCCESS, ptr->status, sl_wksp->data, sl_wksp->len, 0);
 else
   tpreturn(TPFAIL, ptr->status, sl_wksp->data, 0L, 0);
}
```

web_ui.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
*************-*/

/*+
 * Abstract: This file contains the Digital created front end functions
 *                              for the tpcc benchmark.
 *
 * Author: A Bradley & W Carr
 * Creation Date: May 1997
 *
 * Modified history:
 *
 *
*/
#ifdef _WIN32
# include <windows.h>
# include <process.h>
# include <winsock2.h>
# include <sys\timeb.h>
# include <io.h>
#else
  typedef int CRITICAL_SECTION;
# define InitializeCriticalSection( CritSec )
# define EnterCriticalSection( CritSec )
# define LeaveCriticalSection( CritSec )
# define DeleteCriticalSection( CritSec )
# define InterlockedIncrement( pval ) (*(pval))++
# define CopyMemory memcpy
# define _ASSERTE()
#endif
#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.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 <config.h>
#include <ckpt.h>
#include <transpool.h>
#include <tm_util.h>
#include <reg.h>

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


#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_C_STR_2 PT_CREDIT + 1
#define PT_DISC PT_C_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 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 "PAGE_STRMENU_BAR_PAGE_ID"%06d>"
"%55.55s<BR>"
"Select Desired Transaction.<BR>"
MENU_BAR
"</FORM>"END_BODY_STR;

static char szDeliveryFormTemp2i[] =
"<INPUT TYPE=hidden "PAGE_STRDY_RESP_PAGE_ID"######>"
"<PRE>                          Delivery<BR>"
W_ID_STR"#####<BR><BR>"
"Carrier Number: ##<BR><BR>"
EXECUTION_STATUS"Delivery has been queued.<BR>"
DELI_DATA_STR_1
"########## ######## ##### ## #### #### #### #### #### #### ####
#### #### ####"
DELI_DATA_STR_2
"########## ######## ##### ## #### #### #### #### #### #### ####
#### #### ####"
DELI_DATA_STR_2
"</PRE>"
MENU_BAR
"</FORM>"END_BODY_STR;

static char szNewOrderFormTemp2i[] =
"<INPUT TYPE=hidden "PAGE_STRNO_RESP_PAGE_ID"######>"
"<PRE>                         New Order<BR>"
W_ID_STR"#####"
"  "D_ID_STR"##                    Date: ################### <BR>"
"Customer:  ####   Name: ################   Credit: ##   "
"%Disc: #####           <BR>"
O_ID_STR"########  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>"END_BODY_STR;

static char szOrderStatusFormTemp2i[] =
"<INPUT TYPE=hidden "PAGE_STROS_RESP_PAGE_ID"######>"
"<PRE>                          Order-Status<BR>"
W_ID_STR"#####  "D_ID_STR"##<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>"
"</PRE>"
MENU_BAR
"</FORM>"END_BODY_STR;

static char szPaymentFormTemp2i[] =
"<INPUT TYPE=hidden "PAGE_STRPT_RESP_PAGE_ID"######>"
"<PRE>                          Payment<BR>"
"Date: ################### <BR><BR>"
W_ID_STR"#####                  "D_ID_STR"##<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>"END_BODY_STR;

static char szStockLevelFormTemp2i[] =
"<INPUT TYPE=hidden "PAGE_STRSL_RESP_PAGE_ID"######>"
"<PRE>                          Stock-Level<BR>"
W_ID_STR"#####  "D_ID_STR"##<BR><BR>"
"Stock Level Threshold: ##<BR><BR>"
"low stock: ###"
"</PRE>"
MENU_BAR
"</FORM>"END_BODY_STR;

static char szErrorFormTemplate[] =
"<BODY><FORM ACTION=%s METHOD=GET>"
"<INPUT TYPE=hidden "PAGE_STR
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 CheckpointShutdownCmd( EXTENSION_CONTROL_BLOCK
*pECB, int w_id, int ld_id);
void ClearCmd( EXTENSION_CONTROL_BLOCK *pECB );
void ExitCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id );
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);

int  GetCallingThreadLimit( int *pCallingThreadLimit );
void Log( char *szType, char *szStr );
void MakePanicPool( DWORD dwResponseSize );
void MakeResponseHeader( void );
void MakeTemplatePool( DWORD dwFormSize, DWORD
dwResponseSize );
void DeletePanicPool( void );
void DeleteTemplatePool( 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,
                                        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 BOOL                 bLog                        = FALSE;
#ifdef USE_PROCESSES
static char                 szModName[FILENAMESIZE] = "/isapi/web.so";
#else
static char                 szModName[FILENAMESIZE] = { '\0' };
#endif
static char                 szPath[FILENAMESIZE]                = { '\0' };

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 };

#ifdef USE_PROCESSES
  FILE                              *myerr;
#else


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
    *( int * )0 = 1;
#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 );

    if( ERR_SUCCESS != iInitStatus )
      return TRUE;

    break;
  case DLL_THREAD_ATTACH:
    break;
  case DLL_THREAD_DETACH:
    break;
  case DLL_PROCESS_DETACH:

    pTPCCShutdown( );

    pDeleteTransactionPool( );
    DeleteTemplatePool( );
    DeletePanicPool( );

    DeleteCriticalSection( &startupCriticalSection );

    pTPCCCloseLog( );

    break;
  }
  return TRUE;
}
#endif



BOOL WINAPI
GetExtensionVersion(HSE_VERSION_INFO *pVersion)
{

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

  return TRUE;
}



DWORD WINAPI
HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
#pragma message ("FIXME: If the server is set improperly, and the delivery
servers fail to log in in a min TM sybase config, the failure is not reported
until the user tries to log in.")
  int                           Len = sizeof( szPath ) - 1;
  StartupData                   Startup;
  pStartupData                  pStartup = &Startup;
  DWORD                         status;
  int                           dbstatus;
```

```
  int                                   formPoolSize;
  int                                   responsePoolSize;
  int                                   panicPoolSize = 0;
  int                                   transactionPoolSize;
  int                                   transportThreadLimit;
  int                                   callingThreadLimit;
  int                                   transMode;
#ifdef FFE_DEBUG
  unsigned long                         ulHandle;
  unsigned                  uTAddr;
#endif

  if ( ! startupFlag ) {
    EnterCriticalSection( &startupCriticalSection );
    if ( ! startupFlag ) {
#ifdef USE_PROCESSES
  myerr = fopen( "/ffe/bin/myerr", "w" );
  fprintf( myerr, "Running with pid = %d\n", getpid( ));
  fflush( myerr );
  fprintf( myerr, "PathInfo = %s\n", pECB->lpszPathInfo );
  fprintf( myerr, "PathTrans = %s\n", pECB->lpszPathTranslated );
  {
    char                    szTmpFileName[FILENAMESIZE];
    DWORD                   dwTmpFileNameLen = FILENAMESIZE;
    pECB->GetServerVariable( pECB->ConnID, "DOCUMENT_URI",
                                          szTmpFileName,
&dwTmpFileNameLen );
    fprintf( myerr, "DOCUMENT_URI = %s\n", szTmpFileName );
    dwTmpFileNameLen = FILENAMESIZE;
    pECB->GetServerVariable( pECB->ConnID, "DOCUMENT_URL",
                                          szTmpFileName,
&dwTmpFileNameLen );
    fprintf( myerr, "DOCUMENT_URL = %s\n", szTmpFileName );
    dwTmpFileNameLen = FILENAMESIZE;
    pECB->GetServerVariable( pECB->ConnID, "QUERY_STRING",
                                          szTmpFileName,
&dwTmpFileNameLen );
    fprintf( myerr, "QUERY_STRING = %s\n", szTmpFileName );
  }
  fflush( myerr );
#endif
      if( ERR_SUCCESS == iInitStatus ) {
                MakeResponseHeader( );
                pStartup->Path = szPath;
      }
      if( ERR_SUCCESS == iInitStatus ) {
                iInitStatus = GetCallingThreadLimit( &callingThreadLimit );
      }
      if( ERR_SUCCESS == iInitStatus ) {
                iInitStatus = TPCCGetConfig( &bLog,
                                                        &Len, pStartup-
>Path,
                                                        &pStartup->Login,
&pStartup->loginDelay,
                                                        &pStartup->Config,
                                                        &pStartup-
>Transport,
                                                        &pStartup-
>DeliveryTransport );
      }
      if ( ERR_SUCCESS == iInitStatus ) {
                iInitStatus = TPCCTMLoad( );
      }
      if ( ERR_SUCCESS == iInitStatus ) {
                pTPCCOpenLog( szPath );
      }
      if ( ERR_SUCCESS == iInitStatus ) {
                iInitStatus = pTPCCGetTransportMode( &transMode );
      }
      if( ERR_SUCCESS == iInitStatus ) {



                transactionPoolSize = callingThreadLimit;


                pStartup->Transport.asynchronous &= ( TRANS_ASYNC ==
transMode );

                if( pStartup->Transport.generic ) {
                  transportThreadLimit = pStartup->Transport.num_gc;
                }
                else {
                  transportThreadLimit = pStartup->Transport.num_dy +
                    pStartup->Transport.num_no + pStartup->Transport.num_os
+
                    pStartup->Transport.num_pt + pStartup->Transport.num_sl;
                }


                formPoolSize = callingThreadLimit;

                if( pStartup->Transport.asynchronous ) {


                  responsePoolSize = MAX( callingThreadLimit,
transportThreadLimit );


                  panicPoolSize = MAX( callingThreadLimit,
transportThreadLimit );


                  transactionPoolSize += transportThreadLimit;
                }
                else {


                  responsePoolSize = callingThreadLimit;
                  panicPoolSize = callingThreadLimit;
                }


                transactionPoolSize += pStartup-
>DeliveryTransport.num_threads +
                  pStartup->DeliveryTransport.num_queued_deliveries +
                  pStartup->DeliveryTransport.num_queued_responses;

                iWelcomeFormLen =
                  sprintf(szWelcomeForm, szWelcomeFormTemplate,
szModName);
                MakeTemplatePool( formPoolSize, responsePoolSize );
                MakePanicPool( panicPoolSize );
                pMakeTransactionPool( transactionPoolSize );
                pStartup->pTPCCNewOrderResponse =
TPCCNewOrderResponse;
                pStartup->pTPCCOrderStatusResponse =
TPCCOrderStatusResponse;
                pStartup->pTPCCPaymentResponse =
TPCCPaymentResponse;
                pStartup->pTPCCStockLevelResponse =
TPCCStockLevelResponse;
                pStartup->pTPCCResponseComplete =
TPCCResponseComplete;
                dbstatus = pTPCCStartup( pStartup );
                if( ERR_DB_SUCCESS != dbstatus ) {
                  iInitStatus = dbstatus;
                }
#ifdef FFE_DEBUG
                ulHandle = _beginthreadex( NULL, 0, CheckMemory, NULL,
0, &uTAddr );
                _ASSERT( 0 != ulHandle );
#endif
      }
      startupFlag = TRUE;
    }
    LeaveCriticalSection( &startupCriticalSection );
  }

  if( ERR_SUCCESS != iInitStatus )
  {
    if( NULL == gpPanicForms ) {
      MakePanicPool( 50 );
    }
    SendErrorResponse(pECB, iInitStatus, NULL, -1, -1, NULL);
    return HSE_STATUS_SUCCESS;
  }


  if ( bLog )
```

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


  status = ProcessQueryString(pECB);


  return status;
}



void
SendErrorResponse( EXTENSION_CONTROL_BLOCK *pECB, int
iError,
                                char *szMsg, int w_id, int ld_id, pConnData
pConn )
{
  char                                  *szForm;
  int                                   iStrLen;
  char                                  *szDefaultErr = "Unable to look up
this error code.";
  char                                  *szErrorMsg;
  static char                   szNoMsg[] = "";

  if ( !szMsg  )
    szMsg = szNoMsg;

  RESERVE_PANIC_FORM( szForm );

  szErrorMsg = ( NULL == pTPCCErrString ) ?
    szDefaultErr : pTPCCErrString( iError );

  if( NULL != pTPCCErr )
  {
    if( NULL != pConn )
      pTPCCErr( "Transaction error. w_id: %d, ld_id: %d, CC: %I64x, "
                                "status: %d, dbstatus: %d, (%d): %s\r\n",
                                pConn->w_id, pConn->ld_id, pConn->CC,
                                pConn->status, pConn->dbstatus, iError,
szErrorMsg );
    else
      pTPCCErr( "(%d): %s\r\n", iError, szErrorMsg );
  }

  iStrLen = sprintf( szForm, szErrorFormTemplate, szModName,
                                  WDID(w_id,ld_id), iError, 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(szHeader1)-1;
  int                           lpbSize;

  lpbSize = iStrLen + 1;
```

```
 if ( bLog )
   pTPCCLog( "%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, 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, 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, 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, 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 ) {
#pragma message ("FIXME: This is likely the second error report since the
Process*Query functions do their own error reporting!!  This is bad because
we put two error messages on the wire and this could confuse the next
transaction.")
      SendErrorResponse( pECB, retcode, 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, w_id, ld_id );
  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 );
#pragma message ("FIXME: The ordering of the checkpoint commands is
important, because strncmp( 'CheckpointShutdown', 'Checkpoint', strlen(
'Checkpoint' )) is a match.")
  else if( MATCHES_CHECKPOINT_STARTUP( cmdptr ))
    CheckpointStartupCmd( pECB, w_id, ld_id );
  else if( MATCHES_CHECKPOINT_SHUTDOWN( cmdptr ))
    CheckpointShutdownCmd( 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, 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;
}



int
TPCCDeliveryResponse( int retcode, pDeliveryData pDelivery,
                      pDeliveryData
pCompletedDeliveries[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 = (EXTENSION_CONTROL_BLOCK *)pDelivery->CC;

  if ( ERR_DB_PENDING == retcode )
  {
   return( retcode );
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
   SendErrorResponse( pECB, ERR_DELIVERY_NOT_PROCESSED,
NULL,
                                  pDelivery->w_id, pDelivery->ld_id,
                                  (pConnData)pDelivery );
   TRANSACTION_DEBUG_STAGE( pDelivery, UNRESERVING );
   pUnreserveTransactionStruct( DELIVERY_TRANS, &pDelivery );
   for( ii = 0; ii < DELIVERY_RESPONSE_COUNT; ii++ )
   {
    if( NULL != pCompletedDeliveries[ii] )
    {
              TRANSACTION_DEBUG_STAGE(
pCompletedDeliveries[ii], UNRESERVING );
              pUnreserveTransactionStruct( DELIVERY_TRANS,
&pCompletedDeliveries[ii]);
    }
   }
  }
  else          if ( ERR_DB_SUCCESS != retcode )
  {
   SendErrorResponse( pECB, ERR_DB_DELIVERY_NOT_QUEUED,
NULL,
                                  pDelivery->w_id, pDelivery->ld_id,
                                  (pConnData)pDelivery );
   TRANSACTION_DEBUG_STAGE( pDelivery, UNRESERVING );
   pUnreserveTransactionStruct( DELIVERY_TRANS, &pDelivery );
   for( ii = 0; ii < DELIVERY_RESPONSE_COUNT; ii++ )
   {
    if( NULL != pCompletedDeliveries[ii] )
    {
              TRANSACTION_DEBUG_STAGE(
pCompletedDeliveries[ii], UNRESERVING );
              pUnreserveTransactionStruct( DELIVERY_TRANS,
&pCompletedDeliveries[ii]);
    }
   }
  }
  else
 {
   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]);
   TRANSACTION_DEBUG_STAGE( pDelivery, UNRESERVING );
   pUnreserveTransactionStruct( DELIVERY_TRANS, &pDelivery );

   index = D_QUEUE1;
   for( ii = 0; ii < DELIVERY_RESPONSE_COUNT; ii++ ) {
    if( NULL == pCompletedDeliveries[ii] )
             pCompletedDelivery = &blankDelivery;
    else
             pCompletedDelivery = pCompletedDeliveries[ii];
    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( jj = 0; jj < DISTRICTS_PER_WAREHOUSE; jj++ ) {
              PutNumeric(pCompletedDelivery->o_id[jj],
                               deliveryFormIndexesP[index].iLen,

&deliveryForm[deliveryFormIndexesP[index].iStartIndex]);
              index++;
     }
     if( NULL != pCompletedDeliveries[ii] ) {
              TRANSACTION_DEBUG_STAGE(
pCompletedDeliveries[ii], UNRESERVING );
              pUnreserveTransactionStruct( DELIVERY_TRANS,
&pCompletedDeliveries[ii]);
     }
   }

   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 );
 }

 return( retcode );
}



int
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;
 EXTENSION_CONTROL_BLOCK *pECB;

 pECB = (EXTENSION_CONTROL_BLOCK *)pNewOrder->CC;

 if ( ERR_DB_PENDING == retcode )
 {
  return( retcode );
 }
 else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
 {
  SendErrorResponse( pECB, ERR_NEW_ORDER_NOT_PROCESSED,
NULL,
                     pNewOrder->w_id, pNewOrder->ld_id,
                     (pConnData)pNewOrder );
  TRANSACTION_DEBUG_STAGE( pNewOrder, UNRESERVING );
  pUnreserveTransactionStruct( NEW_ORDER_TRANS, &pNewOrder );
  return( retcode );
 }
 else if( ERR_DB_SUCCESS != retcode &&
ERR_DB_NOT_COMMITED != retcode )
 {
  SendErrorResponse( pECB, retcode, NULL,
                     pNewOrder->w_id, pNewOrder->ld_id,
                     (pConnData)pNewOrder );
  TRANSACTION_DEBUG_STAGE( pNewOrder, UNRESERVING );
  pUnreserveTransactionStruct( NEW_ORDER_TRANS, &pNewOrder );
  return( retcode );
 }
 else if ( ERR_DB_SUCCESS == retcode )
 {
  bValid = TRUE;
  execution_status = TRANSACTION_COMMITED;
 }
 else if ( ERR_DB_NOT_COMMITED == retcode )
 {



  retcode = ERR_DB_SUCCESS;
  bValid = FALSE;
  execution_status = ITEM_NUMBER_INVALID;
 }

 RESERVE_RESPONSE( NEW_ORDER_RESPONSE, newOrderForm );

 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]);

 if(bValid)
 {

  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].iStartIn
dex],
            szDate, newOrderResponseIndexes[NO_DATE].iLen);
 }
 else
 {


memcpy(&newOrderForm[newOrderResponseIndexes[NO_DATE].iStartIn
dex],
            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]);
 }
 else
 {



memcpy(&newOrderForm[newOrderResponseIndexes[NO_DISC].iStartInd
ex],
            szBlanks, newOrderResponseIndexes[NO_DISC].iLen);
 }



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

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

 if(bValid)
 {
  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)].iStartInde
x]);
    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)].iStartInde
x]);
    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)].i
StartIndex-1],
                szDollar, 1);
    PutFloat2(pNewOrder->o_ol[i].i_price,

            newOrderResponseIndexes[NO_PRICE+(i*8)].iLen,

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

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

            newOrderResponseIndexes[NO_AMT+(i*8)].iLen,

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

   }

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

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

    for(mm=0; mm<2; mm++)
    {
            memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStart
Index-1],
                szBlanks, newOrderResponseIndexes[jj].iLen+1);
            jj++;
    }
   }
  }
  else
  {
   for(kk=0; kk<3; kk++)
   {

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

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

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

    for(mm=0; mm<2; mm++)
    {
            memcpy(&newOrderForm[newOrderResponseIndexes[jj].iStart
Index-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].iStartI
ndex],
            szBlanks, newOrderResponseIndexes[NO_TOTAL].iLen);
  }
  PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);
  PutHTMLStrings(StrStruct, newOrderForm,
giResponseLen[NEW_ORDER_RESPONSE],
                        &szOutput, &iOutputLen);

  TRANSACTION_DEBUG_STAGE( pNewOrder, UNRESERVING );

  pUnreserveTransactionStruct( NEW_ORDER_TRANS, &pNewOrder );

  SendResponse(pECB, szOutput, iOutputLen);

  UNRESERVE_RESPONSE( NEW_ORDER_RESPONSE, newOrderForm
);

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

  return( retcode );
}



int
TPCCPaymentResponse( int retcode, pPaymentData pPayment )
{
```

```
 char          *ptr;
 char          szcdata[4][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 = (EXTENSION_CONTROL_BLOCK *)pPayment->CC;

 if ( ERR_DB_PENDING == retcode )
 {
  return( retcode );
 }
 else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
 {
  SendErrorResponse( pECB, ERR_PAYMENT_NOT_PROCESSED,
NULL,
                     pPayment->w_id, pPayment->ld_id,
                     (pConnData)pPayment );
  TRANSACTION_DEBUG_STAGE( pPayment, UNRESERVING );
  pUnreserveTransactionStruct( PAYMENT_TRANS, &pPayment );
  return( retcode );
 }
 else if ( ERR_DB_NOT_COMMITED == retcode )
 {
  SendErrorResponse( pECB, ERR_PAYMENT_INVALID_CUSTOMER,
NULL,
                     pPayment->w_id, pPayment->ld_id,
                     (pConnData)pPayment );
  TRANSACTION_DEBUG_STAGE( pPayment, UNRESERVING );
  pUnreserveTransactionStruct( PAYMENT_TRANS, &pPayment );
  return( retcode );
 }
 else if ( ERR_DB_SUCCESS != retcode )
 {
  SendErrorResponse( pECB, retcode, NULL,
                     pPayment->w_id, pPayment->ld_id,
                     (pConnData)pPayment );
  TRANSACTION_DEBUG_STAGE( pPayment, UNRESERVING );
  pUnreserveTransactionStruct( PAYMENT_TRANS, &pPayment );
  return( retcode );
 }

 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].iSt
artIndex],
               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].iStartI
ndex], 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->c_street_2,
                paymentResponseIndexes[PT_C_STR_2].iLen,
                paymentResponseIndexes[PT_C_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].iStartI
ndex],
            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(szcdata[i], ptr, 50);
            szcdata[i][50] = '\0';
   }
   else
            szcdata[i][0] = 0;

   PUT_STRING(szcdata[i],

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].i
StartIndex],
            szBlanks,
paymentResponseIndexes[PT_CUST_DATA+i].iLen);
  }
 }

 PUT_STRING(NULL, 0, 0, StrStruct[ssCnt]);

 PutHTMLStrings(StrStruct, paymentForm,
giResponseLen[PAYMENT_RESPONSE],
                        &szOutput, &iOutputLen);

 TRANSACTION_DEBUG_STAGE( pPayment, UNRESERVING );
 pUnreserveTransactionStruct( PAYMENT_TRANS, &pPayment );

 SendResponse(pECB, szOutput, iOutputLen);

 UNRESERVE_RESPONSE( PAYMENT_RESPONSE, paymentForm );

 if( szOutput != paymentForm )
  UNRESERVE_PANIC_FORM( szOutput );

 return( retcode );
}



int
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 = (EXTENSION_CONTROL_BLOCK *)pOrderStatus->CC;

 if ( ERR_DB_PENDING == retcode )
 {
  return( retcode );
 }
 else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
 {
  SendErrorResponse( pECB,
ERR_ORDER_STATUS_NOT_PROCESSED, NULL,
```

```
                                pOrderStatus->w_id, pOrderStatus->ld_id,
                                (pConnData)pOrderStatus );
    TRANSACTION_DEBUG_STAGE( pOrderStatus, UNRESERVING );
    pUnreserveTransactionStruct( ORDER_STATUS_TRANS,
&pOrderStatus );
    return( retcode );
   }
   else if ( ERR_DB_NOT_COMMITED == retcode )
   {
    SendErrorResponse( pECB, ERR_NOSUCH_CUSTOMER, NULL,
                                pOrderStatus->w_id, pOrderStatus->ld_id,
                                (pConnData)pOrderStatus );
    TRANSACTION_DEBUG_STAGE( pOrderStatus, UNRESERVING );
    pUnreserveTransactionStruct( ORDER_STATUS_TRANS,
&pOrderStatus );
    return( retcode );
   }
   else if ( ERR_DB_SUCCESS != retcode )
   {
    SendErrorResponse( pECB, retcode, NULL,
                                pOrderStatus->w_id, pOrderStatus->ld_id,
                                (pConnData)pOrderStatus );
    TRANSACTION_DEBUG_STAGE( pOrderStatus, UNRESERVING );
    pUnreserveTransactionStruct( ORDER_STATUS_TRANS,
&pOrderStatus );
    return( retcode );
   }

   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].iStart
Index],
                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)].iStartIn
dex]);
    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)].iStartInde
x]);

memcpy(&orderStatusForm[orderStatusResponseIndexes[OS_AMT+(i*5)].i
StartIndex-1],
                szDollar, 1);
    PutFloat2(pOrderStatus->s_ol[i].ol_amount,
                orderStatusResponseIndexes[OS_AMT+(i*5)].iLen,

&orderStatusForm[orderStatusResponseIndexes[OS_AMT+(i*5)].iStartInde
x]);
    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<MAX_OL; 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);

   TRANSACTION_DEBUG_STAGE( pOrderStatus, UNRESERVING );
   pUnreserveTransactionStruct( ORDER_STATUS_TRANS, &pOrderStatus
);

   SendResponse(pECB, szOutput, iOutputLen);

   UNRESERVE_RESPONSE( ORDER_STATUS_RESPONSE,
orderStatusForm );
```

```
  if( szOutput != orderStatusForm )
    UNRESERVE_PANIC_FORM( szOutput );

  return( retcode );
}



int
TPCCStockLevelResponse( int retcode, StockLevelData *pStockLevel )
{
  char *stockLevelForm;
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = (EXTENSION_CONTROL_BLOCK *)pStockLevel->CC;

  if ( ERR_DB_PENDING == retcode )
  {
    return( retcode );
  }
  else if ( ERR_DB_DEADLOCK_LIMIT == retcode )
  {
    SendErrorResponse( pECB, ERR_STOCKLEVEL_NOT_PROCESSED,
NULL,
                       pStockLevel->w_id, pStockLevel->ld_id,
                       (pConnData)pStockLevel );
    TRANSACTION_DEBUG_STAGE( pStockLevel, UNRESERVING );
    pUnreserveTransactionStruct( STOCK_LEVEL_TRANS, &pStockLevel
);
    return( retcode );
  }
  else if ( ERR_DB_SUCCESS != retcode )
  {
    SendErrorResponse( pECB, retcode, NULL,
                       pStockLevel->w_id, pStockLevel->ld_id,
                       (pConnData)pStockLevel );
    TRANSACTION_DEBUG_STAGE( pStockLevel, UNRESERVING );
    pUnreserveTransactionStruct( STOCK_LEVEL_TRANS, &pStockLevel
);
    return( retcode );
  }

  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]);

  TRANSACTION_DEBUG_STAGE( pStockLevel, UNRESERVING );
  pUnreserveTransactionStruct( STOCK_LEVEL_TRANS, &pStockLevel );

  SendResponse(pECB, stockLevelForm,
               giResponseLen[STOCK_LEVEL_RESPONSE]);

  UNRESERVE_RESPONSE( STOCK_LEVEL_RESPONSE,
stockLevelForm );

  return( retcode );
}



void
TPCCResponseComplete( CallersContext CC )
{
  DWORD   status;
  EXTENSION_CONTROL_BLOCK *pECB;

  pECB = (EXTENSION_CONTROL_BLOCK *)CC;
  status = HSE_STATUS_SUCCESS_AND_KEEP_CONN;
  (pECB->ServerSupportFunction)(pECB->ConnID,
HSE_REQ_DONE_WITH_SESSION,
                                &status, NULL, NULL);
}

int
ParseDeliveryQuery( char *lpszQueryString, pDeliveryData pDelivery )
{
  char                  *ptr;
  char
          *deliveryVals[MAXDELIVERYVALS];

  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 < 1 ||
       pDelivery->o_carrier_id > DISTRICTS_PER_WAREHOUSE )
    return ERR_DELIVERY_CARRIER_ID_RANGE;

  return( ERR_SUCCESS );
}

int
ParseStockLevelQuery( char *lpszQueryString, pStockLevelData
pStockLevel )
{
  char                  *ptr;
  char
          *stockLevelVals[MAXSTOCKLEVELVALS];

  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;

  return( ERR_SUCCESS );
}



int
ProcessDeliveryQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                      int w_id, int ld_id )
{
  int                   retcode;
  pDeliveryData         pDelivery;
  pDeliveryData
          pCompletedDeliveries[DELIVERY_RESPONSE_COUNT];

  pDelivery = pReserveTransactionStruct( DELIVERY_TRANS );
```

```
  pDelivery->w_id = w_id;
  pDelivery->ld_id = ld_id;
  pDelivery->CC = (CallersContext)pECB;

  if ( ERR_SUCCESS != ( retcode = ParseDeliveryQuery( lpszQueryString,

pDelivery )))
  {
    pUnreserveTransactionStruct( DELIVERY_TRANS, &pDelivery );
    return( retcode );
  }

  TRANSACTION_DEBUG_STAGE( pDelivery, CALLING_LH );

  retcode = pTPCCDelivery( &pDelivery, pCompletedDeliveries );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
#endif

  TRANSACTION_DEBUG_STAGE( pDelivery, CALLING_RESP );

  retcode = TPCCDeliveryResponse( retcode, pDelivery,
pCompletedDeliveries );

  return( retcode );
}


int
ProcessNewOrderQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                      int w_id, int ld_id )
{
  int                   retcode;
  NewOrderData          *pNewOrder;

  pNewOrder = pReserveTransactionStruct( NEW_ORDER_TRANS );

  pNewOrder->w_id = w_id;
  pNewOrder->ld_id = ld_id;
  pNewOrder->CC = (CallersContext)pECB;

  if ( ERR_SUCCESS != ( retcode = ParseNewOrderQuery(
lpszQueryString,

pNewOrder )))
  {
    pUnreserveTransactionStruct( NEW_ORDER_TRANS, &pNewOrder );
    return( retcode );
  }

  TRANSACTION_DEBUG_STAGE( pNewOrder, CALLING_LH );

  retcode = pTPCCNewOrder( &pNewOrder );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
#endif

  TRANSACTION_DEBUG_STAGE( pNewOrder, CALLING_RESP );

  retcode = TPCCNewOrderResponse( retcode, pNewOrder );

  return( retcode );
}


int
ProcessOrderStatusQuery( EXTENSION_CONTROL_BLOCK *pECB,
char *lpszQueryString,
                         int w_id, int ld_id )
{
  int                   retcode;
  OrderStatusData       *pOrderStatus;

  pOrderStatus = pReserveTransactionStruct( ORDER_STATUS_TRANS );

  pOrderStatus->w_id = w_id;
  pOrderStatus->ld_id = ld_id;
  pOrderStatus->CC = (CallersContext)pECB;

  if( ERR_SUCCESS != ( retcode = ParseOrderStatusQuery(
lpszQueryString,

                pOrderStatus )))
  {
    pUnreserveTransactionStruct( ORDER_STATUS_TRANS,
&pOrderStatus );
    return( retcode );
  }

  TRANSACTION_DEBUG_STAGE( pOrderStatus, CALLING_LH );

  retcode = pTPCCOrderStatus( &pOrderStatus );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
#endif

  TRANSACTION_DEBUG_STAGE( pOrderStatus, CALLING_RESP );

  retcode = TPCCOrderStatusResponse( retcode, pOrderStatus );

  return( retcode );
}


int
ProcessPaymentQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                     int w_id, int ld_id )
{
  int                   retcode;
  PaymentData           *pPayment;

  pPayment = pReserveTransactionStruct( PAYMENT_TRANS );

  pPayment->w_id = w_id;
  pPayment->ld_id = ld_id;
  pPayment->CC = (CallersContext)pECB;

  if( ERR_SUCCESS != ( retcode = ParsePaymentQuery( lpszQueryString,

pPayment ))
  {
    pUnreserveTransactionStruct( PAYMENT_TRANS, &pPayment );
    return( retcode );
  }

  TRANSACTION_DEBUG_STAGE( pPayment, CALLING_LH );

  retcode = pTPCCPayment( &pPayment );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
#endif

  TRANSACTION_DEBUG_STAGE( pPayment, CALLING_RESP );

  retcode = TPCCPaymentResponse( retcode, pPayment );

  return( retcode );
}


int
ProcessStockLevelQuery( EXTENSION_CONTROL_BLOCK *pECB, char
*lpszQueryString,
                        int w_id, int ld_id )
{
  int                   retcode;
  StockLevelData        *pStockLevel;

  pStockLevel = pReserveTransactionStruct( STOCK_LEVEL_TRANS );
```

```
  pStockLevel->w_id = w_id;
  pStockLevel->ld_id = ld_id;
  pStockLevel->CC = (CallersContext)pECB;

  if ( ERR_SUCCESS != ( retcode = ParseStockLevelQuery(
lpszQueryString,

                pStockLevel )))
  {
    pUnreserveTransactionStruct( STOCK_LEVEL_TRANS, &pStockLevel
);
    return( retcode );
  }

  TRANSACTION_DEBUG_STAGE( pStockLevel, CALLING_LH );

  retcode = pTPCCStockLevel( &pStockLevel );

#ifdef FFE_DEBUG
  _ASSERT(VALID_DB_ERR(retcode));
#endif

  TRANSACTION_DEBUG_STAGE( pStockLevel, CALLING_RESP );

  retcode = 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 );


  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
BeginCmd( EXTENSION_CONTROL_BLOCK *pECB )
{
  SendWelcomeForm(pECB);
}



void
CheckpointCmd(EXTENSION_CONTROL_BLOCK *pECB, int w_id, int
ld_id )
{
  int                                   retcode;
  CheckpointData            *pCheckpoint;

  pCheckpoint = pReserveTransactionStruct( CHECKPOINT_TRANS );

  pCheckpoint->w_id = w_id;
  pCheckpoint->ld_id = ld_id;
  pCheckpoint->CC = (CallersContext)pECB;

  retcode = pTPCCCheckpoint( &pCheckpoint );
  if ( ERR_DB_SUCCESS == retcode ) {
    SendMainMenuForm(pECB, w_id, ld_id,
                     "Checkpoint issued (non-blocking), completed
(blocking).");
  }
  else {
    SendErrorResponse( pECB, retcode, NULL, w_id, ld_id, NULL );
  }

  pUnreserveTransactionStruct( CHECKPOINT_TRANS, &pCheckpoint );
}



void
CheckpointShutdownCmd(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id )
{
  int                                   retcode;
  pConnData                 pConn;

  pConn = pReserveTransactionStruct( DISCONNECT_TRANS );

  pConn->w_id = w_id;
  pConn->ld_id = ld_id;
  pConn->CC = (CallersContext)pECB;

  if( ERR_DB_SUCCESS != ( retcode = pTPCCCheckpointDisconnect(
&pConn )))
  {
    SendErrorResponse( pECB, retcode, NULL, w_id, ld_id, NULL );
  }
  else
  {
    SendMainMenuForm(pECB, w_id, ld_id, "Checkpoint Shutdown
Succeded.");
  }

  pUnreserveTransactionStruct( DISCONNECT_TRANS, &pConn );
}



void
```

```
CheckpointStartupCmd(EXTENSION_CONTROL_BLOCK *pECB, int
w_id, int ld_id )
{
  int                                   retcode;
  pLoginData                            pLogin;

  pLogin = pReserveTransactionStruct( CONNECT_TRANS );


  pLogin->w_id = w_id;
  pLogin->ld_id = ld_id;
  pLogin->CC = (CallersContext)pECB;

  if( ERR_SUCCESS != ( retcode = GetLoginDataCheckpoint( pLogin )) ||
      ERR_DB_SUCCESS != ( retcode = pTPCCCheckpointConnect(
&pLogin )))
  {
    SendErrorResponse( pECB, retcode, NULL, w_id, ld_id, NULL );
  }
  else
  {
    SendMainMenuForm(pECB, w_id, ld_id, "Checkpoint Startup
Succeded.");
  }

  pUnreserveTransactionStruct( CONNECT_TRANS, &pLogin );
}



void
ClearCmd(EXTENSION_CONTROL_BLOCK *pECB)
{
  pTPCCCloseLog( );
  pTPCCOpenLog( szPath );

  SendWelcomeForm(pECB);
}



void
ExitCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id, int ld_id )
{
  ConnData                      *pConn;

  pConn = pReserveTransactionStruct( DISCONNECT_TRANS );

  pConn->w_id = w_id;
  pConn->ld_id = ld_id;
  pConn->CC = (CallersContext)pECB;

  pTPCCDisconnect( &pConn );

  pUnreserveTransactionStruct( DISCONNECT_TRANS, &pConn );

  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 *pLogin;
  char            *ptr;

  if ( !GetCharKeyValuePtr( pECB->lpszQueryString, '4', &ptr ) ||
      ( 0 == ( *w_id = atoi( ptr ))) ||
      ( *w_id < 0 ))
  {
    SendErrorResponse( pECB, ERR_W_ID_INVALID, NULL, *w_id, -1,
NULL );
    goto SubmitDone;
  }

  if ( !GetCharKeyValuePtr( pECB->lpszQueryString, '5', &ptr ) ||
      ( 0 == ( *ld_id = atoi( ptr ))) ||
      ( *ld_id > 10 ) ||
      ( *ld_id < 0 ))
  {
    SendErrorResponse( pECB, ERR_D_ID_INVALID, NULL, *w_id,
*ld_id, NULL );
    goto SubmitDone;
  }

  pLogin = pReserveTransactionStruct( CONNECT_TRANS );

  pLogin->w_id = *w_id;
  pLogin->ld_id = *ld_id;
  pLogin->CC = (CallersContext)pECB;

  if( ERR_SUCCESS != ( iStatus = GetLoginData( pLogin )) ||
      ERR_DB_SUCCESS != ( iStatus = pTPCCConnect( &pLogin ) ))
  {
    SendErrorResponse( pECB, iStatus, NULL, *w_id, *ld_id, NULL );
  }
  else
  {
    SendMainMenuForm(pECB, *w_id, *ld_id, NULL);
  }

  pUnreserveTransactionStruct( CONNECT_TRANS, &pLogin );

SubmitDone:
  return;
}

#ifdef FFE_DEBUG


void
MemoryCheckCmd( EXTENSION_CONTROL_BLOCK *pECB, int w_id,
int ld_id )
{
  _ASSERTE( _CrtCheckMemory( ));

  SendErrorResponse( pECB, ERR_SUCCESS, NULL, w_id, ld_id, NULL
);
}
#endif


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;
}

int
GetCallingThreadLimit( int *pCallingThreadLimit )
{
#ifdef USE_PROCESSES
 *pCallingThreadLimit = 1;
 return( ERR_SUCCESS );
#else
  int                                   Status;
  int                                   size;

 size = sizeof( *pCallingThreadLimit );
 Status = GetRegValue( INETINFO_CLASS, "PoolThreadLimit",
                                        &size, (char *)pCallingThreadLimit );
 if( ERR_SUCCESS != Status )
 {
  return ERR_CANT_FIND_POOLTHREADLIMIT;
 }
 return( ERR_SUCCESS );
#endif
}

#ifdef FFE_DEBUG


unsigned __stdcall
CheckMemory(void *param)
{
 while (TRUE)
 {
  _ASSERTE( _CrtCheckMemory( ));
  Sleep( 1000 );
 }
 return 0;
}

#endif
```

web_ui.h

```
#ifndef WEB_UI_H
#define WEB_UI_H
/*+************************************************************
**************
 *                                                            *
 *  COPYRIGHT (c) 1997, 2000 BY                               *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                              *
 *                                                            *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                            *
 *                                                            *

************************************************************
************-*/

/*+
 * 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:
 *
 *
 */

#ifdef WEB_UI_C
#define WEBGLOBAL(thing,initializer) thing = initializer
#else
#define WEBGLOBAL(thing,initializer) extern thing
#endif


BOOL GetNumeric(char *ptr, int *iValue);
BOOL GetValuePtr(char *pProcessedQuery[], int iIndex, char **pValue);



#define DID 0
#define CID DID+1

#define CLT_O CID+1
#define MAXORDERSTATUSVALS CLT_O + 1

#define TT  0
#define MAXSTOCKLEVELVALS TT + 1

#define QUEUETIME 0
#define OCD 1
#define MAXDELIVERYVALS OCD + 1

#define CWI   CID + 1
#define CDI   CWI + 1
#define CLT_P CDI + 1
#define HAM   CLT_P + 1
#define MAXPAYMENTVALS HAM + 1

#define SP00  CID + 1
#define IID00 SP00 + 1
#define QTY00 IID00 + 1
#define SP01  QTY00 + 1
#define IID01 SP01 + 1
#define QTY01 IID01 + 1
#define SP02  QTY01 + 1
#define IID02 SP02 + 1
#define QTY02 IID02 + 1
#define SP03  QTY02 + 1
#define IID03 SP03 + 1
#define QTY03 IID03 + 1
#define SP04  QTY03 + 1
#define IID04 SP04 + 1
#define QTY04 IID04 + 1
#define SP05  QTY04 + 1
#define IID05 SP05 + 1
#define QTY05 IID05 + 1
#define SP06  QTY05 +1
#define IID06 SP06 + 1
#define QTY06 IID06 + 1
#define SP07  QTY06 + 1
#define IID07 SP07 + 1
#define QTY07 IID07 + 1
#define SP08  QTY07 + 1
#define IID08 SP08 + 1
#define QTY08 IID08 + 1
#define SP09  QTY08 + 1
#define IID09 SP09 + 1
#define QTY09 IID09 + 1
#define SP10  QTY09 + 1
#define IID10 SP10 + 1
#define QTY10 IID10 + 1
#define SP11  QTY10 + 1
#define IID11 SP11 + 1
#define QTY11 IID11 + 1
#define SP12  QTY11 + 1
#define IID12 SP12 + 1
#define QTY12 IID12 + 1
#define SP13  QTY12 + 1
#define IID13 SP13 + 1
#define QTY13 IID13 + 1
#define SP14  QTY13 + 1
#define IID14 SP14 + 1
#define QTY14 IID14 + 1
#define MAXNEWORDERVALS QTY14 + 1


#define CONNECT_APPL_STR \
"GET %s?0=Begin HTTP/1.0\nConnection: Keep-Alive\n\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 GET_DY_MENU_STR \
"GET %s?3=d%d&0=Delivery HTTP/1.0\nConnection: Keep-Alive\n\n"
#define GET_DY_TXN_STR \
"GET %s?3=D%d&6=%d&7=%d&0=Process HTTP/1.0\nConnection:
Keep-Alive\n\n"


#define GET_NO_MENU_STR \
"GET %s?3=n%d&0=NewOrder HTTP/1.0\nConnection: Keep-Alive\n\n"
#define GET_NO_TXN_STR1 \
"GET %s?3=N%d&8=%d&9=%d&"
#define GET_NO_TXN_STR2 \
"0=Process HTTP/1.0\nConnection: Keep-Alive\n\n"


#define GET_OS_MENU_STR \
"GET %s?3=o%d&0=OrderStatus HTTP/1.0\nConnection: Keep-Alive\n\n"
```

```
#define GET_OS_ID_TXN_STR \
"GET %s?3=O%d&8=%d&9=&Y=%s&0=Process HTTP/1.0\nConnection:
Keep-Alive\n\n"
#define GET_OS_NAME_TXN_STR \
"GET %s?3=O%d&8=%d&9=%d&Y=&0=Process HTTP/1.0\nConnection:
Keep-Alive\n\n"


#define GET_PT_MENU_STR \
"GET %s?3=p%d&0=Payment HTTP/1.0\nConnection: Keep-Alive\n\n"
#define GET_PT_ID_TXN_STR \
"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"
#define GET_PT_NAME_TXN_STR \
"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"


#define GET_SL_MENU_STR \
"GET %s?3=s%d&0=StockLevel HTTP/1.0\nConnection: Keep-Alive\n\n"
#define GET_SL_TXN_STR \
"GET %s?3=S%d&x=%d&0=Process HTTP/1.0\nConnection: Keep-
Alive\n\n"


#define CP_STARTUP_STR \
"GET %s?3=W%d&4=1&5=1&0=CheckpointStartup
HTTP/1.0\nConnection: Keep-Alive\n\n"
#define CP_RUN_STR \
"GET %s?3=W%d&4=1&5=1&0=Checkpoint HTTP/1.0\nConnection:
Keep-Alive\n\n"
#define CP_SHUTDOWN_STR \
"GET %s?3=W%d&4=1&5=1&0=CheckpointShutdown
HTTP/1.0\nConnection: Keep-Alive\n\n"

#define EXECUTION_STATUS "Execution Status: "
#define TRANSACTION_COMMITED "Transaction commited."
#define ITEM_NUMBER_INVALID "Item number is not valid."

#define DELI_DATA_STR_1 "Previous Deliveries:<BR>"
#define DELI_DATA_STR_2 "<BR>"

#define W_ID_STR "Warehouse: "
#define D_ID_STR "District: "
#define O_ID_STR "Order Number: "

#define ERROR_VALUE_STR "Error: "


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

#define PANIC_FORM_SIZE 4096




#define PAGE_STR "NAME=3 VALUE="
#define PAGE_ID_LEN 1

#define INVALID_PAGE_ID                 "\0"
#define GREETING_PAGE_ID        "W"
#define MENU_BAR_PAGE_ID        "M"
#define CKPT_STARTUP_PAGE_IDMENU_BAR_PAGE_ID
#define CKPT_PAGE_ID                    MENU_BAR_PAGE_ID
#define CKPT_SHUTDOWN_PAGE_ID           MENU_BAR_PAGE_ID
#define ERROR_PAGE_ID                   "e"
#define DY_FORM_PAGE_ID                 "D"
#define DY_RESP_PAGE_ID                 "d"
#define NO_FORM_PAGE_ID                 "N"
#define NO_RESP_PAGE_ID                 "n"
#define OS_FORM_PAGE_ID                 "O"
#define OS_RESP_PAGE_ID                 "o"
#define PT_FORM_PAGE_ID                 "P"
#define PT_RESP_PAGE_ID                 "p"
#define SL_FORM_PAGE_ID                 "S"
#define SL_RESP_PAGE_ID                 "s"

#define END_BODY_STR "</BODY>"

#define MENU_SYNC_STR END_BODY_STR
#define GREETING_SYNC_STR END_BODY_STR
#define NO_SYNC_STR   END_BODY_STR
#define PT_SYNC_STR   END_BODY_STR
#define DY_SYNC_STR   END_BODY_STR
#define SL_SYNC_STR   END_BODY_STR
#define OS_SYNC_STR   END_BODY_STR
#define CP_SYNC_STR   END_BODY_STR
#define CP_STARTUP_SYNC_STR MENU_SYNC_STR
#define CP_SHUTDOWN_SYNC_STR MENU_SYNC_STR

#ifdef API_WEB_C

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=&",
};
#endif

#if 0
#define
PARSE_QUERY_STRING(pQueryString,varMax,charTable,valTable)\
                {\
                  int ii;\
                  char *ptr, *tmpPtr;\
                  ptr = pQueryString;\
                  for (ii=0; ii < varMax; ii++)\
                  {\
                    if ( !(tmpPtr=strstr(ptr, stringTable[ii])) )\
                      valTable[ii] = NULL;\
                    else\
                    {\
                      ptr = tmpPtr;\
                      if ( !(ptr=strchr(ptr, '=')) )\
                                valTable[ii] = NULL;\
                      else\
                                valTable[ii] = ++ptr;\
                    }\
                  }\
                }
#else
#define
PARSE_QUERY_STRING(pQueryString,varMax,charTable,valTable)\
{\
  int ii;\
  char *ptr;\
  int iKey;\
  ptr = pQueryString;\
  for( ii=0; ii<varMax; ii++ ) {\
```

```
    iKey = charTable[ii];\
    valTable[ii] = NULL;\
    if( iKey == *ptr && '=' == *++ptr ) {\
      valTable[ii] = ++ptr;\
    }\
    while( *ptr ) {\
      if( '&' == *ptr ) {\
                ptr++;\
                break;\
      }\
      ptr++;\
    }\
  }\
}
#endif

typedef struct _FORMINDEXES
{
  int iStartIndex;
  int iLen;
} FORM_INDEXES;

WEBGLOBAL(FORM_INDEXES deliveryFormIndexesI[4], { 0 });
WEBGLOBAL(FORM_INDEXES deliveryFormIndexesP[33], { 0 });
WEBGLOBAL(FORM_INDEXES newOrderFormIndexes[4], { 0 });
WEBGLOBAL(FORM_INDEXES newOrderResponseIndexes[136], { 0 });
WEBGLOBAL(FORM_INDEXES orderStatusFormIndexes[4], { 0 });
WEBGLOBAL(FORM_INDEXES orderStatusResponseIndexes[88], { 0 });
WEBGLOBAL(FORM_INDEXES paymentFormIndexes[4], { 0 });
WEBGLOBAL(FORM_INDEXES paymentResponseIndexes[38], { 0 });
WEBGLOBAL(FORM_INDEXES stockLevelFormIndexes[5], { 0 });
WEBGLOBAL(FORM_INDEXES stockLevelResponseIndexes[7], { 0 });

#ifdef WEB_UI_C
char deliveryStrs[] = {'6', '7'};
char newOrderStrs[] = {
'8', '9',
'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'};
char orderStatusStrs[] = {'8', '9', 'Y'};
char paymentStrs[] = {'8', '9', 'Z', 'v', 'Y', 'w'};
char stockLevelStrs[] = {'x'};
#else
extern char deliveryStrs[];
extern char newOrderStrs[];
extern char orderStatusStrs[];
extern char paymentStrs[];
extern char stockLevelStrs[];
#endif

#endif
```

webd.c

```
/*+**********************************************************
**************
 *                                                          *
 * COPYRIGHT (c) 1999, 2000 BY                              *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 * 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
DIGITAL EQUIPMENT   *
 * CORPORATION.                                             *
 *                                                          *
 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS   *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *

**********************************************************
*************-*/

#ifdef _WIN32
# include <windows.h>
# include <process.h>
# include <direct.h>
# define PATH_MAX MAX_PATH
#else
# include <limits.h>
# include <critsecunix.h>
# include <wintypes.h>
#endif

#include <stdio.h>
#include <stdarg.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/stat.h>

#include <netserver.h>
#include <service.h>
#include <dynlib.h>

#ifdef _WIN32
# include <httpext.h>
# include <crtdbg.h>
#else
  typedef struct _ConnStruct *pConnStruct;
# define HCONN pConnStruct
# define HSE_STATUS_ERROR 0
# define HSE_STATUS_SUCCESS 1
# define HSE_STATUS_SUCCESS_AND_KEEP_CONN 2
# define HSE_STATUS_PENDING 3
# define HSE_REQ_DONE_WITH_SESSION 4
# define HSE_REQ_SEND_RESPONSE_HEADER 5
  typedef struct
  { DWORD dwExtensionVersion; char lpszExtensionDesc[256]; }
HSE_VERSION_INFO;
  typedef BOOL ( *PFN_GETEXTENSIONVERSION )(
HSE_VERSION_INFO *pVersion );
  typedef BOOL ( *PFN_GETSERVERVARIABLE )( HCONN      hConn,
                                                    LPSTR
lpszVariableName,
                                                    LPVOID
lpvBuffer,

LPDWORD     lpdwSize );

  typedef BOOL ( *PFN_WRITECLIENT )( HCONN
              ConnID,
                                                    LPVOID
              Buffer,
                                                    LPDWORD
              lpdwBytes,
                                                    DWORD
              dwReserved );
  typedef BOOL ( *PFN_READCLIENT )( HCONN      ConnID,
```

```
                                        LPVOID
lpvBuffer,
                                        LPDWORD
lpdwSize );
  typedef BOOL ( *PFN_SERVERSUPPORTFUNCTION )( HCONN
hConn,

DWORD      dwHSERequest,

LPVOID     lpvBuffer,

LPDWORD    lpdwSize,

LPDWORD    lpdwDataType );
  typedef struct
  { char *lpszQueryString;
    HCONN ConnID;
    PFN_GETSERVERVARIABLE GetServerVariable;
    PFN_READCLIENT ReadClient;
    PFN_SERVERSUPPORTFUNCTION ServerSupportFunction;
    PFN_WRITECLIENT WriteClient;
  } EXTENSION_CONTROL_BLOCK;
  typedef DWORD ( *PFN_HTTPEXTENSIONPROC )(
EXTENSION_CONTROL_BLOCK *pECB );
#endif

#ifdef FFE_DEBUG
static int tmpDbgFlag;
#endif


#define SZSERVICENAME        "webd"

#define SZSERVICEDISPLAYNAME "webd"


#define WEBD_STATUS_PROCESSING 0

#define DEFAULT_THREAD_COUNT 16
#define WEBPORT 81
#define MAX_LOADED_DLLS 10

#define VERSION_MAJOR 0
#define VERSION_MINOR 1
#define SOFTWARE_VERSION "Digital TPCC V0.1"
#define SOFTWARE_VERSION_LEN strlen( SOFTWARE_VERSION )
#define COPYRIGHT "Copyright Digital Equipment Corp. 1996"

#define ERROR_LOADING "ERROR - ISAPI dll failed to load."

#define HSE_STATUS_TO_NSSTAT( HSEStatus, NSStat ) \
if( HSE_STATUS_SUCCESS_AND_KEEP_CONN == ( HSEStatus )) (
NSStat ) = NS_SUCCESS;\
else if( HSE_STATUS_SUCCESS == ( HSEStatus )) ( NSStat ) =
NS_CLOSE;\
else if( HSE_STATUS_ERROR == ( HSEStatus )) ( NSStat ) =
NS_FAILURE;\
else if( HSE_STATUS_PENDING == ( HSEStatus )) ( NSStat ) =
NS_PENDING;\
else _ASSERT( FALSE );

typedef struct _DLLRec {
  char                                  dll_name[PATH_MAX];
  PFN_GETEXTENSIONVERSION   pGetExtensionVersion;
  PFN_HTTPEXTENSIONPROC                 pHttpExtensionProc;
  HINSTANCE                             hInstance;
} DLLRec;

static int       num_loaded_dlls = 0;
static DLLRec              DLLRecs[MAX_LOADED_DLLS];

#define SCRIPTS "/scripts"
#define SCRIPTS_LEN strlen( SCRIPTS )
#define SLASH "/"
#define SLASH_LEN strlen( SLASH )
#define IISADMIN "/iisadmin"
#define IISADMIN_LEN strlen( IISADMIN )

typedef struct _VirtualRoots {
  char                                  slash[PATH_MAX];
  char                                  iisadmin[PATH_MAX];
  char                                  scripts[PATH_MAX];
} VirtualRoots;

VirtualRoots               gVirtualRoots;

#define REMOTE_USER "REMOTE_USER"
#define REMOTE_USER_LEN strlen( REMOTE_USER )
#define REMOTE_ADDR "REMOTE_ADDR"
#define REMOTE_ADDR_LEN strlen( REMOTE_ADDR )
#define SERVER_NAME "SERVER_NAME"
#define SERVER_NAME_LEN strlen( SERVER_NAME )
#define SERVER_SOFTWARE "SERVER_SOFTWARE"
#define SERVER_SOFTWARE_LEN strlen( SERVER_SOFTWARE )

static DWORD webPort = WEBPORT;
static DWORD threadCount = 0;
static NSHANDLE                       gnsHandle;
static int debugFlag=0;
static LONG client_id = 0;

typedef struct _ConnStruct {
  struct _ConnStruct       *next;
  struct _ConnStruct       *prev;
  int                                   client_id;
  DWORD                                 status;
  EXTENSION_CONTROL_BLOCK ECB;
  DLLRec                   *pdll_rec;
  BOOL                                  keep_alive;


  char                                  buf[4096];
  char                                  *lpszHTTPLoc;
  char                                  *lpszPathLoc;
  char                                  *lpszQuesLoc;
  char                                  *lpszCommand;
  char                                  *lpszLineBegin;
  char                                  *lpszNextLineBegin;
  char                                  *lpszLineEnd;
  char                                  *lpszBuf;
  int                                   iLineLen;
  BOOL                                  bEOL;
  BOOL                                  bEndOfRequest;
  BOOL                                  bGotCommand;
  BOOL                                  bGotHTTP;
  char                                  lpszPath[PATH_MAX];
  BOOL                                  bCGIRequest;
} ConnStruct;

CRITICAL_SECTION                        ConnListCritSec;
ConnStruct                              *pConnList = NULL;


void parseCommandLine( int argc, char* argv[] );

#if LINK_TPCC
  BOOL APIENTRY ACMSxpDllMain(HANDLE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved);
  BOOL APIENTRY DllMain(HANDLE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved);
#endif


void
debug( int id, char* message, ... )
{
  char                          str[4096];
  va_list       list;

  if( debugFlag ) {
    va_start( list, message );
    vsprintf( str, message, list );

    fprintf( stderr, "[%d] %s\n", id, str );
  }
}

_
static BOOL
GetConfig( void )
```

```
{
#ifdef _WIN32
  HKEY                          hKey;
  BYTE                          szTmp[256];
  DWORD                         type;
  DWORD                         size;
  char                          *ptr;
  LONG                          status;
  BOOL                          retval;


  status = RegOpenKeyEx(HKEY_LOCAL_MACHINE,
            "SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Parameters\
\Virtual Roots",
                                0, KEY_READ, &hKey);
  if( ERROR_SUCCESS != status )
  {
    return( FALSE );
  }

  retval = TRUE;

  if( retval )
  {
    size = sizeof(szTmp);
    status = RegQueryValueEx(hKey, "/", 0, &type, szTmp, &size);
    if( ERROR_SUCCESS != status ) {
      size = sizeof(szTmp);
      status = RegQueryValueEx(hKey, "/,", 0, &type, szTmp, &size);
      if( ERROR_SUCCESS != status )
      {
                retval = FALSE;
      }
    }
  }
  if( retval )
  {
    if ( NULL != ( ptr = strchr( szTmp, ',' )))
      *ptr = '\0';
    strcpy(gVirtualRoots.slash, szTmp);
  }

  if( retval )
  {
    size = sizeof(szTmp);
    status = RegQueryValueEx(hKey, "/iisadmin", 0, &type, szTmp, &size);
    if( ERROR_SUCCESS != status ) {
      size = sizeof(szTmp);
      status = RegQueryValueEx(hKey, "/iisadmin,", 0, &type, szTmp,
&size);
      if( ERROR_SUCCESS != status )
      {
                retval = FALSE;
      }
    }
  }
  if( retval )
  {
    if ( NULL != ( ptr = strchr( szTmp, ',' )))
      *ptr = '\0';
    strcpy(gVirtualRoots.iisadmin, szTmp);
  }

  if( retval )
  {
    size = sizeof(szTmp);
    status = RegQueryValueEx(hKey, "/Scripts", 0, &type, szTmp, &size);
    if( ERROR_SUCCESS != status ) {
      size = sizeof(szTmp);
      status = RegQueryValueEx(hKey, "/Scripts,", 0, &type, szTmp, &size);
      if( ERROR_SUCCESS != status )
      {
                retval = FALSE;
      }
    }
  }
  if( retval )
  {
    if ( NULL != ( ptr = strchr( szTmp, ',' )))
      *ptr = '\0';
    strcpy(gVirtualRoots.scripts, szTmp);
  }

  RegCloseKey(hKey);

  if( !retval )
  {
    return( retval );
  }

  status = RegOpenKeyEx( HKEY_LOCAL_MACHINE,

            "SYSTEM\\CurrentControlSet\\Services\\InetInfo\\Parameters",
                                0, KEY_READ, &hKey );
  if( ERROR_SUCCESS != status )
  {
    return FALSE;
  }

  size = sizeof( threadCount );
  status = RegQueryValueEx( hKey, "PoolThreadLimit", 0, &type,
                                (char *)&threadCount, &size );
  if( ERROR_SUCCESS != status ) {
    threadCount = DEFAULT_THREAD_COUNT;
  }

  RegCloseKey(hKey);

#endif
  return TRUE;
}


void
error( int id, char* message, int ex )
{
  fprintf(stderr, "*** [%d] ERROR: %s\n", id, message);
  if(ex) {
#pragma message ("FIXME: How do we tell the SCM (if we are running as
a service) that we had an error and want to exit?")

  }
}


DLLRec *
LoadDLL( ConnStruct *pcs, char *dll_name )
{
  int                           ii;
  DLLRec                *pdll_rec;
  char                          *end_dir1;
  char                          *end_dir2;
  char                          dir[PATH_MAX];


  for( ii = 0; ii < num_loaded_dlls; ii++ ) {
    if( 0 == strcmp( dll_name, DLLRecs[ii].dll_name )) {

      return( &DLLRecs[ii] );
    }
  }


  if( MAX_LOADED_DLLS == num_loaded_dlls ) {

    return NULL;
  }


  pdll_rec = &DLLRecs[num_loaded_dlls];
  strcpy( pdll_rec->dll_name, dll_name );


  strcpy( dir, dll_name );
  end_dir1 = strrchr( dir, '\\' );
  end_dir2 = strrchr( dir, '/' );
  if( end_dir1 > end_dir2 )
    *end_dir1 = '\0';
  else
    *end_dir2 = '\0';
```

```
  chdir( dir );

  pdll_rec->hInstance = DynamicLibraryLoad( pdll_rec->dll_name );

  if( NULL == pdll_rec->hInstance )
  {
    debug( pcs->client_id, "Load failed err = %s", DynamicLibraryError( ));
    return NULL;
  }
  if( NULL == ( pdll_rec->pGetExtensionVersion =
(PFN_GETEXTENSIONVERSION)
                              DynamicLibrarySymbol( pdll_rec->hInstance,
"GetExtensionVersion" ))) {
    return NULL;
  }
  if( NULL == ( pdll_rec->pHttpExtensionProc =
(PFN_HTTPEXTENSIONPROC)
                              DynamicLibrarySymbol( pdll_rec->hInstance,
"HttpExtensionProc" ))) {
    return NULL;
  }
  num_loaded_dlls++;

  return pdll_rec;
}

BOOL WINAPI GetServerVariable( HCONN      hConn,
                                          LPSTR      lpszVariableName,
                                          LPVOID     lpvBuffer,
                                          LPDWORD    lpdwSize )
{
  fprintf( stderr, "Called GetServerVariable()\n" );

#if 0
  struct sockaddr_in        addr;
  int                               len;
  char                              *lpszAddr;
  ConnStruct                *pcs = hConn;

  if( 0 == strncmp( lpszVariableName, REMOTE_USER,
REMOTE_USER_LEN )) {
    strcpy( lpvBuffer, "Administrator" );
    *lpdwSize = strlen( "Administrator" );
    return TRUE;
  }
  else if( 0 == strncmp( lpszVariableName, REMOTE_ADDR,
REMOTE_ADDR_LEN )) {
    len = sizeof( addr );
    if( SOCKET_ERROR == getpeername( pcs->s, (struct sockaddr *)&addr,
&len ))
      return FALSE;
    if( NULL == ( lpszAddr = inet_ntoa( addr.sin_addr )))
      return FALSE;
    strcpy( lpvBuffer, lpszAddr );
    *lpdwSize = strlen( lpszAddr );
    return TRUE;
  }
  else if( 0 == strncmp( lpszVariableName, SERVER_NAME,
SERVER_NAME_LEN )) {
    if( NULL == lpvBuffer ) {
      *lpdwSize = 0;
      return TRUE;
    }
    if( SOCKET_ERROR == gethostname( lpvBuffer, *lpdwSize ))
      return FALSE;
    *lpdwSize = strlen( lpvBuffer );
    return TRUE;
  }
  else if(0==strncmp( lpszVariableName, SERVER_SOFTWARE,
SERVER_SOFTWARE_LEN)) {
    strcpy( lpvBuffer, SOFTWARE_VERSION );
    *lpdwSize = SOFTWARE_VERSION_LEN;
    *lpdwSize = 0;
    return TRUE;
  }
#endif
  return FALSE;
}

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

  static char nulls[6] = {0,0,0,0,0,0};

  pcs = ConnID;

  if( debugFlag ) {
    fprintf( stderr, "Sending: " );
    fwrite( Buffer, *lpdwBytes, 1, stderr );
    fprintf( stderr, "\n" );
  }

  SendToNetClient( pcs, Buffer, *lpdwBytes );

  return TRUE;
}

BOOL WINAPI ReadClient( HCONN      ConnID,
                                      LPVOID     lpvBuffer,
                                      LPDWORD    lpdwSize )
{
  fprintf( stderr, "Called ReadClient\n" );

  return FALSE;
}

BOOL WINAPI ServerSupportFunction( HCONN      hConn,
                                                  DWORD
dwHSERequest,
                                                  LPVOID     lpvBuffer,
                                                  LPDWORD
lpdwSize,
                                                  LPDWORD
lpdwDataType )
{
  char *lpszDataType;
  char *lpszBuffer;
  ConnStruct *pcs;
  char szTmp[4096];
  char date[29];
  int iTmpLen;
  struct tm *newtime;
  time_t aclock;
  DWORD                             status;
  DWORD                             oldStatus;
  NSSTAT                    NSStat;

  pcs = hConn;

  if( HSE_REQ_DONE_WITH_SESSION == dwHSERequest ) {
    status = *(DWORD *)lpvBuffer;



    oldStatus = InterlockedExchange(( LPLONG )&pcs->status, ( LONG
)status );

    if(( WEBD_STATUS_PROCESSING != oldStatus ) &&
      ( HSE_STATUS_PENDING != oldStatus ))
    {


      _ASSERT( FALSE );
    }

    HSE_STATUS_TO_NSSTAT( status, NSStat );

    SendToNetClientComplete( pcs, NSStat );
  }
  else if( HSE_REQ_SEND_RESPONSE_HEADER == dwHSERequest ) {
    time( &aclock );
    newtime = localtime( &aclock );
    memcpy( date, asctime( newtime ), 24 );
```

```
    date[24] = ' ';
    date[25] = ' ';
    date[26] = ' ';
    date[27] = ' ';
    date[28] = '\0';

    lpszDataType = (char *)lpdwDataType;
    lpszBuffer = (char *)lpvBuffer;
    iTmpLen =  sprintf( szTmp, "HTTP/1.0 %s\r\n"
                                        "Server: Digital TPCC
V%d.%d\r\nDate: %s\r\n"
                                        "%s",
                                        lpszBuffer, VERSION_MAJOR,
VERSION_MINOR,
                                        date, lpszDataType );


    if( debugFlag ) {
      fprintf( stderr, "Support sending: %s", szTmp );
    }

    SendToNetClient( pcs, szTmp, iTmpLen );
  }
  else {

    dwHSERequest;
    _ASSERT( FALSE );
  }
  return TRUE;
}
_

static void
BadRequest( ConnStruct *pcs, char *szStr, ... )
{
  int           lpbSize;
  int           iSize;
  char          szHeader[128];
  char          szHeader1[128];
  char                                  szTmp1[4096];
  char                                  szTmp2[4096];
  int                                   length;
  va_list                       list;


  va_start( list, szStr );
  length = vsprintf( szTmp1, szStr, list );
  va_end( list );

  length = sprintf( szTmp2, "<HTML>%s</HTML>", szTmp1 );

  lpbSize = length + 1;

  iSize = sprintf(szHeader, "400 Bad Request");
  sprintf( szHeader1,
                "Content-Type: text/html\r\n"
                "Content-Length: %d\r\n"
                "\r\n", lpbSize );

  ServerSupportFunction(pcs, HSE_REQ_SEND_RESPONSE_HEADER,
szHeader,
                                        &iSize, (LPDWORD)szHeader1);
  WriteClient(pcs, szTmp2, &lpbSize, 0);

  return;
}
_


DWORD
executeCGIQuery( ConnStruct *pcs )
{
  DWORD                                 dwStatus;

  if(debugFlag)
    fprintf(stderr, "[%d] Executing '%s'...\n",
                pcs->client_id, pcs->ECB.lpszQueryString);


  dwStatus = (*pcs->pdll_rec->pHttpExtensionProc)( &pcs->ECB );

  if( HSE_STATUS_SUCCESS == dwStatus ) {
    debug( pcs->client_id, "Done");
  }
  else if( HSE_STATUS_SUCCESS_AND_KEEP_CONN == dwStatus ) {
    debug( pcs->client_id, "Done");
  }
  else {
    debug( pcs->client_id,  "status returned != SUCCESS" );
  }
  return dwStatus;
}
_

DWORD
returnPage( ConnStruct *pcs )
{
  FILE                                  *fp;
  size_t                        len;
#define BUFSIZE 4096
  char                                  buf[BUFSIZE];
  struct stat                   sbuf;
  char                                  szHeader1[100];
  static char                   szHeader[] = "200 Ok";
  static int                    iSize = sizeof(szHeader)-1;
  int                                   lpbSize;

  if( -1 == stat( pcs->ECB.lpszQueryString, &sbuf )) {
    BadRequest( pcs, "Stat call failed on requested page." );
    return HSE_STATUS_ERROR;
  }

  if( NULL == ( fp = fopen( pcs->ECB.lpszQueryString, "rb" ))) {
    BadRequest( pcs, "Open call failed on requested page." );
    return HSE_STATUS_ERROR;
  }

  lpbSize = sbuf.st_size;

  if( pcs->keep_alive )
    sprintf( szHeader1,
                "Connection: keep-alive\r\n"
                "Content-Type: text/html\r\n"
                "Content-Length: %d\r\n"
                "\r\n", lpbSize );
  else
    sprintf( szHeader1,
                "Content-Type: text/html\r\n"
                "Content-Length: %d\r\n"
                "\r\n", lpbSize );

  ServerSupportFunction( pcs, HSE_REQ_SEND_RESPONSE_HEADER,
                                        szHeader, &iSize,
(LPDWORD)szHeader1 );

  do {
    len = fread( buf, 1, BUFSIZE, fp );
    if( BUFSIZE == len ) {
      WriteClient( pcs, buf, &len, 0 );
    }
    else if( 0 < len ) {
      WriteClient( pcs, buf, &len, 0 );
      if( feof( fp ))
                return HSE_STATUS_SUCCESS;
      else
                return HSE_STATUS_ERROR;
    }
    else {
      if( feof( fp ))
                return HSE_STATUS_SUCCESS;
      else
                return HSE_STATUS_ERROR;
    }
  } while( 1 );
}
_


NSSTAT
RecvFromNetClient( LPVOID lpvApplData,
                                LPBYTE lpBytes, DWORD dwCount,
```

```
                                LPBYTE *lppBytes, LPDWORD lpdwSize )
{
 ConnStruct              *pcs;
 int                             ii;
 BOOL                            GoAgain;
 DWORD                           status;
 DWORD                           oldStatus;
 NSSTAT                  NSStat;

 pcs = lpvApplData;

 if( 0 == dwCount ) {
  if( HSE_STATUS_SUCCESS != pcs->status ) {


   error( pcs->client_id, "connection closed.", 0);
  }
#pragma message ("FIXME: What do we do with 0 bytes data?")
  _ASSERT( FALSE );
 }
 else {
  pcs->lpszBuf += dwCount;
  *pcs->lpszBuf = '\0';
 }


 status = HSE_STATUS_SUCCESS_AND_KEEP_CONN;
 do {
#pragma message ("FIXME: Logic of this routine needs to be reviewed for
cases where the connection should be closed.")
  while( !pcs->bEndOfRequest ) {
   pcs->bEOL = FALSE;
   while( !pcs->bEOL ) {
            if( NULL != ( pcs->lpszLineEnd = strchr( pcs->lpszLineBegin,
'\n' ))) {
             pcs->bEOL = TRUE;
             pcs->lpszNextLineBegin = pcs->lpszLineEnd + 1;
             *pcs->lpszLineEnd = '\0';
             if( pcs->lpszLineEnd > pcs->lpszLineBegin ) {
              if( '\r' == *( pcs->lpszLineEnd - 1 )) {
               pcs->lpszLineEnd--;
               *pcs->lpszLineEnd = '\0';
              }
             }
             pcs->iLineLen = pcs->lpszLineEnd - pcs->lpszLineBegin;
            }
            else {

             *lppBytes = pcs->lpszBuf;
             *lpdwSize = sizeof( pcs->buf ) - ( pcs->lpszBuf - pcs->buf );
             HSE_STATUS_TO_NSSTAT( status, NSStat );
             return( NSStat );
#if 0
             buflen = recv( pcs->s, pcs->lpszBuf,
                                    ( sizeof(pcs->buf) - ( pcs->lpszBuf -
pcs->buf )), 0 );
             if( SOCKET_ERROR == buflen ) {
              error( pcs->client_id, "Socket recv failure.", 0);
              break;
             }
             else if( 0 == buflen ) {
              if( HSE_STATUS_SUCCESS != pcs->status ) {


               error( pcs->client_id, "connection closed.", 0);
              }
              break;
             }
             else {
              pcs->lpszBuf += buflen;
              *pcs->lpszBuf = '\0';
             }
#endif
            }
   }
   if( 0 == pcs->iLineLen ) {

            pcs->bEndOfRequest = TRUE;
   }
   if( !pcs->bGotCommand ) {

            if( 0 != strncmp( pcs->lpszLineBegin, "GET", 3 ) ||
               ' ' != pcs->lpszLineBegin[3] ) {
             pcs->bGotCommand = FALSE;
             BadRequest( pcs, pcs->lpszLineBegin );
             status = HSE_STATUS_ERROR;
             break;
            }

            pcs->lpszPathLoc = pcs->lpszLineBegin + 4;


            pcs->lpszHTTPLoc = strstr( pcs->lpszPathLoc, " HTTP/1.0" );
            if( pcs->lpszHTTPLoc ) {
             pcs->bGotHTTP = TRUE;
             *pcs->lpszHTTPLoc++ = '\0';
             pcs->bEndOfRequest = FALSE;
            }
            else {
             pcs->lpszHTTPLoc = strstr( pcs->lpszPathLoc, " HTTP/1.1" );
             if( pcs->lpszHTTPLoc ) {
              pcs->bGotHTTP = TRUE;
              *pcs->lpszHTTPLoc++ = '\0';
              pcs->bEndOfRequest = FALSE;
             }
             else {
              pcs->bEndOfRequest = TRUE;
             }
            }


            if( NULL != ( pcs->lpszQuesLoc = strchr( pcs->lpszPathLoc, '?'
))) {

             pcs->bCGIRequest = TRUE;
             *pcs->lpszQuesLoc = '\0';
             pcs->lpszCommand = pcs->lpszQuesLoc+1;
            }


            for( ii = 0; '\0' != pcs->lpszPathLoc[ii]; ii++ )
             pcs->lpszPathLoc[ii] = tolower( pcs->lpszPathLoc[ii] );
            if( 0 == strncmp( pcs->lpszPathLoc, SCRIPTS, SCRIPTS_LEN
) &&
              ('/' == pcs->lpszPathLoc[SCRIPTS_LEN] ||
               '\\' == pcs->lpszPathLoc[SCRIPTS_LEN] )) {
             strcpy( pcs->lpszPath, gVirtualRoots.scripts );
             pcs->lpszPathLoc += SCRIPTS_LEN;
            }
            else if( 0 == strncmp( pcs->lpszPathLoc, IISADMIN,
IISADMIN_LEN ) &&
                        ('/' == pcs->lpszPathLoc[IISADMIN_LEN] ||
                         '\\' == pcs->lpszPathLoc[IISADMIN_LEN] )) {
             strcpy( pcs->lpszPath, gVirtualRoots.iisadmin );
             pcs->lpszPathLoc += IISADMIN_LEN;
            }
            else
             strcpy( pcs->lpszPath, gVirtualRoots.slash );


            strcat( pcs->lpszPath, pcs->lpszPathLoc );

            if( pcs->bCGIRequest ) {

             if( NULL == ( pcs->pdll_rec = LoadDLL( pcs, pcs->lpszPath
))) {
              BadRequest( pcs, ERROR_LOADING );
              status = HSE_STATUS_ERROR;
              break;
             }
            }

            pcs->bGotCommand = TRUE;
   }
   else if( !pcs->bEndOfRequest ) {
            char                            *lpchar;
            for( lpchar = pcs->lpszLineBegin; '\0' != *lpchar; lpchar++ ) {
             *lpchar = tolower( *lpchar );
            }
            if( strstr( pcs->lpszLineBegin, "keep-alive" )) {
```

```
                pcs->keep_alive = TRUE;
              }
    }
    pcs->lpszLineBegin = pcs->lpszNextLineBegin;
   }
   if(debugFlag)
    fprintf(stderr, "[%d] Query: '%s'\n", pcs->client_id, pcs->buf);


   if( pcs->bGotCommand ) {

    pcs->status = WEBD_STATUS_PROCESSING;
    if( pcs->bCGIRequest ) {
              if(debugFlag)
                fprintf(stderr, "[%d] Command: '%s'\n",
                        pcs->client_id, pcs->lpszCommand);
              pcs->ECB.lpszQueryString = pcs->lpszCommand;
              status = executeCGIQuery( pcs );
              pcs->bCGIRequest = FALSE;
    }
    else {
              pcs->ECB.lpszQueryString = pcs->lpszPath;
              status = returnPage( pcs );
              if( HSE_STATUS_SUCCESS == status && pcs->keep_alive )
              {
                status = HSE_STATUS_SUCCESS_AND_KEEP_CONN;
              }

    }

    GoAgain = (pcs->bGotCommand &&
                        (
HSE_STATUS_SUCCESS_AND_KEEP_CONN == status ||
                          HSE_STATUS_PENDING == status ) &&
                        pcs->keep_alive );
    oldStatus =
              ( DWORD )InterlockedCompareExchange(( PVOID *)&pcs-
>status,
                                                  ( PVOID
)status,
                                                  ( PVOID
)WEBD_STATUS_PROCESSING );
    _ASSERT(( WEBD_STATUS_PROCESSING == oldStatus ) ||
                ( HSE_STATUS_SUCCESS == oldStatus ) ||
                ( HSE_STATUS_SUCCESS_AND_KEEP_CONN ==
oldStatus ) ||
                ( HSE_STATUS_ERROR == oldStatus ));
    pcs->bGotCommand = FALSE;
    pcs->buf[0] = '\0';
    pcs->lpszBuf = pcs->buf;
    pcs->lpszLineBegin = pcs->buf;
    pcs->bEndOfRequest = FALSE;
    pcs->keep_alive = FALSE;
   }

  } while( GoAgain );

  HSE_STATUS_TO_NSSTAT( status, NSStat );
  return( NSStat );
}
_

VOID
NetClientStarting( LPVOID lpvApplData, LPBYTE *lppBytes, LPDWORD
lpdwSize )
{
  ConnStruct                    *pcs;

  InterlockedIncrement( &client_id );
  debug( client_id, "Client connected!");

  pcs = lpvApplData;


  EnterCriticalSection( &ConnListCritSec );
  pcs->next = pConnList;
  if( NULL != pConnList )
    pConnList->prev = pcs;
  pConnList = pcs;
  pcs->prev = NULL;
  LeaveCriticalSection( &ConnListCritSec );

  pcs->client_id = client_id;
  pcs->ECB.GetServerVariable = GetServerVariable;
  pcs->ECB.WriteClient = WriteClient;
  pcs->ECB.ReadClient = ReadClient;
  pcs->ECB.ServerSupportFunction = ServerSupportFunction;
  pcs->ECB.ConnID = pcs;
  pcs->keep_alive = FALSE;


  pcs->bCGIRequest = FALSE;
  pcs->bGotCommand = FALSE;
  pcs->buf[0] = '\0';
  pcs->lpszBuf = pcs->buf;
  pcs->lpszLineBegin = pcs->buf;
  pcs->bEndOfRequest = FALSE;
  pcs->keep_alive = FALSE;

  *lppBytes = pcs->lpszBuf;
  *lpdwSize = sizeof( pcs->buf );
}
_

VOID
NetClientStopping( LPVOID lpvApplData )
{
#pragma message ("FIXME: Need cleanup code to respond to server telling
us the client has exited.")
  ConnStruct                    *pcs;

  pcs = lpvApplData;

  debug( pcs->client_id, "Client exiting!");


  EnterCriticalSection( &ConnListCritSec );

  if( NULL != pcs->prev )
    pcs->prev->next = pcs->next;
  else
    pConnList = pcs->next;

  if( NULL != pcs->next )
    pcs->next->prev = pcs->prev;

  LeaveCriticalSection( &ConnListCritSec );
}
_


void
parseCommandLine( int argc, char* argv[] )
{
  char                                  *BinaryPathName;
  int                                   ii;

  for(ii = 1; ii < argc; ii++) {
    if( 0 == strcmp(argv[ii], "-d") )
      debugFlag = 1;
    else if( 0 == strncmp(argv[ii], "-p", 2) ) {
      if( '\0' != argv[ii][2] )
                webPort = atoi( &argv[ii][2] );
      else
                webPort = atoi( argv[++ii] );
    }
    else if( 0 == strcmp(argv[ii], "-v") ) {
      fprintf(stderr, "\nwebd - web daemon for Windows NT, version
%d.%d\n%s\n",
                VERSION_MAJOR, VERSION_MINOR, COPYRIGHT);
      exit(0);
    }
    else if( 0 == strcmp(argv[ii], "-?") ) {
      BinaryPathName = argv[0];


      printf( "%s -d                    print debugging messages\n",
                BinaryPathName );
      printf( "%s -p <port>             run on port <port>\n",
                BinaryPathName );
```

```
    printf( "%s -v                    print version\n",
                    BinaryPathName );
   }
   else
    fprintf(stderr, "Ignoring unknown option '%s'...\n", argv[ii]);
  }
}
_

void
ServiceNames( pServiceNameInfo pNames )
{
  pNames->ServiceName = SZSERVICENAME;
  pNames->DisplayName = SZSERVICEDISPLAYNAME;
  pNames->Dependencies = "\0";
}
_

VOID
ServiceStart( DWORD dwArgc, LPTSTR *lpszArgv )
{
  int                                   server_id = 0;

#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);
#endif

#ifdef DEBUG_ENTRY
  _ASSERT( FALSE );
#endif

  parseCommandLine( dwArgc, lpszArgv );

  InitializeCriticalSection( &ConnListCritSec );

  if( !GetConfig( ))
    error( 0,  "error getting web server configuration.", 1 );

  gnsHandle = InitializeNetServer( webPort, threadCount, sizeof( ConnStruct
));

#if LINK_TPCC
  ACMSxpDllMain(NULL, DLL_PROCESS_ATTACH, NULL);
  DllMain(NULL, DLL_PROCESS_ATTACH, NULL);
#endif

#ifdef _WIN32
  ReportStatusToSCMgr( SERVICE_RUNNING, 0, NO_ERROR, 0 );
#endif

  debug( server_id, "Ready for connections...");

  AcceptNetClients( );

#if LINK_TPCC
  DllMain(NULL, DLL_PROCESS_DETACH, NULL);
  ACMSxpDllMain(NULL, DLL_PROCESS_DETACH, NULL);
#endif
}
_

VOID
ServiceStop( VOID )
{



  TerminateNetServer( gnsHandle );

  DeleteCriticalSection( &ConnListCritSec );
}
```

Appendix B

4600_devices.create

```
disk init
  name = 'ordlne01',
  physname = '/tpcc_devs/ordlne01',
  vdevno = 1,
  size = 7360000
go
disk init
  name = 'ordlne02',
  physname = '/tpcc_devs/ordlne02',
  vdevno = 2,
  size = 7360000
go
disk init
  name = 'ordlne03',
  physname = '/tpcc_devs/ordlne03',
  vdevno = 3,
  size = 7360000
go
disk init
  name = 'ordlne04',
  physname = '/tpcc_devs/ordlne04',
  vdevno = 4,
  size = 7360000
go
disk init
  name = 'ordlne05',
  physname = '/tpcc_devs/ordlne05',
  vdevno = 5,
  size = 7360000
go
disk init
  name = 'ordlne06',
  physname = '/tpcc_devs/ordlne06',
  vdevno = 6,
  size = 7360000
go
disk init
  name = 'ordlne07',
  physname = '/tpcc_devs/ordlne07',
  vdevno = 7,
  size = 7360000
go
disk init
  name = 'ordlne08',
  physname = '/tpcc_devs/ordlne08',
  vdevno = 8,
  size = 7360000
go
disk init
  name = 'orders01',
  physname = '/tpcc_devs/orders01',
  vdevno = 9,
  size = 980992
go
disk init
  name = 'orders02',
  physname = '/tpcc_devs/orders02',
  vdevno = 10,
  size = 980992
go
disk init
  name = 'orders03',
  physname = '/tpcc_devs/orders03',
  vdevno = 11,
  size = 980992
go
disk init
  name = 'history01',
  physname = '/tpcc_devs/history01',
  vdevno = 12,
  size = 5888000
go
disk init
  name = 'cidx01',
  physname = '/tpcc_devs/cidx01',
  vdevno = 13,
  size = 1177600
go
disk init
  name = 'cidx02',
  physname = '/tpcc_devs/cidx02',
  vdevno = 14,
  size = 1177600
go
disk init
  name = 'cidx03',
  physname = '/tpcc_devs/cidx03',
  vdevno = 15,
  size = 1177600
go
disk init
  name = 'stock01',
  physname = '/tpcc_devs/stock01',
  vdevno = 16,
  size = 3014656
go
disk init
  name = 'stock02',
  physname = '/tpcc_devs/stock02',
  vdevno = 17,
  size = 3014656
go
disk init
  name = 'stock03',
  physname = '/tpcc_devs/stock03',
  vdevno = 18,
  size = 3014656
go
disk init
  name = 'stock04',
  physname = '/tpcc_devs/stock04',
  vdevno = 19,
  size = 3014656
go
disk init
  name = 'stock05',
  physname = '/tpcc_devs/stock05',
  vdevno = 20,
  size = 3014656
go
disk init
  name = 'stock06',
  physname = '/tpcc_devs/stock06',
  vdevno = 21,
  size = 3014656
go
disk init
  name = 'stock07',
  physname = '/tpcc_devs/stock07',
  vdevno = 22,
  size = 3014656
go
disk init
  name = 'stock08',
  physname = '/tpcc_devs/stock08',
  vdevno = 23,
  size = 3014656
go
disk init
  name = 'stock09',
  physname = '/tpcc_devs/stock09',
  vdevno = 24,
  size = 3014656
go
disk init
  name = 'stock10',
  physname = '/tpcc_devs/stock10',
  vdevno = 25,
  size = 3014656
go
disk init
```

```
  name = 'stock11',
  physname = '/tpcc_devs/stock11',
  vdevno = 26,
  size = 3014656
go
disk init
  name = 'stock12',
  physname = '/tpcc_devs/stock12',
  vdevno = 27,
  size = 3014656
go
disk init
  name = 'stock13',
  physname = '/tpcc_devs/stock13',
  vdevno = 28,
  size = 3014656
go
disk init
  name = 'stock14',
  physname = '/tpcc_devs/stock14',
  vdevno = 29,
  size = 3014656
go
disk init
  name = 'stock15',
  physname = '/tpcc_devs/stock15',
  vdevno = 30,
  size = 3014656
go
disk init
  name = 'stock16',
  physname = '/tpcc_devs/stock16',
  vdevno = 31,
  size = 3014656
go
disk init
  name = 'stock17',
  physname = '/tpcc_devs/stock17',
  vdevno = 32,
  size = 3014656
go
disk init
  name = 'stock18',
  physname = '/tpcc_devs/stock18',
  vdevno = 33,
  size = 3014656
go
disk init
  name = 'stock19',
  physname = '/tpcc_devs/stock19',
  vdevno = 34,
  size = 3014656
go
disk init
  name = 'stock20',
  physname = '/tpcc_devs/stock20',
  vdevno = 35,
  size = 3014656
go
disk init
  name = 'stock21',
  physname = '/tpcc_devs/stock21',
  vdevno = 36,
  size = 3014656
go
disk init
  name = 'stock22',
  physname = '/tpcc_devs/stock22',
  vdevno = 37,
  size = 3014656
go
disk init
  name = 'stock23',
  physname = '/tpcc_devs/stock23',
  vdevno = 38,
  size = 3014656
go
disk init
  name = 'stock24',
  physname = '/tpcc_devs/stock24',
  vdevno = 39,
  size = 3014656
go
disk init
  name = 'stock25',
  physname = '/tpcc_devs/stock25',
  vdevno = 40,
  size = 3014656
go
disk init
  name = 'cust01',
  physname = '/tpcc_devs/cust01',
  vdevno = 41,
  size = 3210752
go
disk init
  name = 'cust02',
  physname = '/tpcc_devs/cust02',
  vdevno = 42,
  size = 3210752
go
disk init
  name = 'cust03',
  physname = '/tpcc_devs/cust03',
  vdevno = 43,
  size = 3210752
go
disk init
  name = 'cust04',
  physname = '/tpcc_devs/cust04',
  vdevno = 44,
  size = 3210752
go
disk init
  name = 'cust05',
  physname = '/tpcc_devs/cust05',
  vdevno = 45,
  size = 3210752
go
disk init
  name = 'cust06',
  physname = '/tpcc_devs/cust06',
  vdevno = 46,
  size = 3210752
go
disk init
  name = 'cust07',
  physname = '/tpcc_devs/cust07',
  vdevno = 47,
  size = 3210752
go
disk init
  name = 'cust08',
  physname = '/tpcc_devs/cust08',
  vdevno = 48,
  size = 3210752
go
disk init
  name = 'cust09',
  physname = '/tpcc_devs/cust09',
  vdevno = 49,
  size = 3210752
go
disk init
  name = 'cust10',
  physname = '/tpcc_devs/cust10',
  vdevno = 50,
  size = 3210752
go
disk init
  name = 'cust11',
  physname = '/tpcc_devs/cust11',
  vdevno = 51,
  size = 3210752
go
disk init
  name = 'cust12',
  physname = '/tpcc_devs/cust12',
  vdevno = 52,
  size = 3210752
```

```
go
disk init
  name = 'cust13',
  physname = '/tpcc_devs/cust13',
  vdevno = 53,
  size = 3210752
go
disk init
  name = 'cust14',
  physname = '/tpcc_devs/cust14',
  vdevno = 54,
  size = 3210752
go
disk init
  name = 'cust15',
  physname = '/tpcc_devs/cust15',
  vdevno = 55,
  size = 3210752
go
disk init
  name = 'log01',
  physname = '/tpcc_devs/log01',
  vdevno = 56,
  size = 15360000
go
create database tpcc
on master = 2500, ordlne01 = 14375
, ordlne02 = 14375
, ordlne03 = 14375
, ordlne04 = 14375
, ordlne05 = 14375
, ordlne06 = 14375
, ordlne07 = 14375
, ordlne08 = 14375
, orders01 = 1916
, orders02 = 1916
, orders03 = 1916
, history01 = 11500
, cidx01 = 2300
, cidx02 = 2300
, cidx03 = 2300
, stock01 = 5888
, stock02 = 5888
, stock03 = 5888
, stock04 = 5888
, stock05 = 5888
, stock06 = 5888
, stock07 = 5888
, stock08 = 5888
, stock09 = 5888
, stock10 = 5888
, stock11 = 5888
, stock12 = 5888
, stock13 = 5888
, stock14 = 5888
, stock15 = 5888
, stock16 = 5888
, stock17 = 5888
, stock18 = 5888
, stock19 = 5888
, stock20 = 5888
, stock21 = 5888
, stock22 = 5888
, stock23 = 5888
, stock24 = 5888
, stock25 = 5888
, cust01 = 6271
, cust02 = 6271
, cust03 = 6271
, cust04 = 6271
, cust05 = 6271
, cust06 = 6271
, cust07 = 6271
, cust08 = 6271
, cust09 = 6271
, cust10 = 6271
, cust11 = 6271
, cust12 = 6271
, cust13 = 6271
, cust14 = 6271
, cust15 = 6271

log on log01 = 30000
go
use tpcc
go
sp_addsegment Scache , tpcc , master
go
sp_addsegment Scidx , tpcc , cidx01
go
sp_extendsegment Scidx , tpcc ,  cidx02
go
sp_extendsegment Scidx , tpcc ,  cidx03
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_extendsegment Sordlne , tpcc ,  ordlne05
go
sp_extendsegment Sordlne , tpcc ,  ordlne06
go
sp_extendsegment Sordlne , tpcc ,  ordlne07
go
sp_extendsegment Sordlne , tpcc ,  ordlne08
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
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 , cidx03
go
sp_dropsegment 'system', tpcc , cidx03
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 , ordlne05
go
sp_dropsegment 'system', tpcc , ordlne05
go
sp_dropsegment 'default', tpcc , ordlne06
go
sp_dropsegment 'system', tpcc , ordlne06
go
sp_dropsegment 'default', tpcc , ordlne07
go
sp_dropsegment 'system', tpcc , ordlne07
go
sp_dropsegment 'default', tpcc , ordlne08
go
sp_dropsegment 'system', tpcc , ordlne08
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
use master
go
checkpoint
go
```

bulk_sybase.c

```
#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;
}

typedef struct
{
        char *terminator;
        int termlen;
        int type;
} bind_parm;

bind_parm parm[MAX_T] =
{
                {NULL, 0, SYBINT4},
                {NULL, 0, SYBINT4},
                {NULL, 0, SYBFLT8},
                {NULL, 0, SYBFLT8},
                {"",   1, SYBCHAR},
                {NULL, 0, SYBDATETIME},
                {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;


        for (db=0; db<MAXOPENS; db++)
                if (dbproc[db] == NULL) break;
        count[db] = 0;


        dbmsghandle(msg_handler);
```

```
        dberrhandle(err_handler);


        if (dbinit() != SUCCEED)
                printf("Can't initialize the DB library\n");


        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);


        DBSETLPACKET(login, 4096);


        dbproc[db] = dbopen(login, NULL);
        if (dbproc[db] == NULL)
                printf("Can't establish connection. Is DSQUERY
set?\n");


        if (database != NULL)
                if (dbuse(dbproc[db], database) != SUCCEED)
                        printf("Can't select database: %s\n",
database);


        dbloginfree(login);


        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);
}
```

error.c

```
#if ! lint
static char *sddsId = "@(#)  error.c  1.1  4/30/91  19:47:32";
#endif


#include <stdio.h>
#ifdef _NTINTEL
#include <stdlib.h>
#include <windows.h>
#endif


#include <sybfront.h>
#include <sybdb.h>


#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;
{


      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(-100);
}
#else
int
err_handler(dbproc, severity, errno, oserr)
  DBPROCESS *dbproc;
  int severity;
  int errno;
  int oserr;
{

      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(-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;
{


        if (msgno == DUMB_MESSAGE || msgno == ABORT_ERROR ||
msgno ==
5703 || msgno == 5704 || msgno == 4843)
                return(SUCCEED);


        if (msgno == 1205)
        {

                *((DBBOOL *) dbgetuserdata(dbproc)) = TRUE;


#ifdef _NTINTEL
                Sleep((DWORD) 2000);
#else
                sleep((unsigned) 2);
#endif
                return(SUCCEED);

        }

        fprintf(stderr, "msg no %d -\n%s", msgno, msgtext);


        exit(-101);
}
```

load.c

```
typedef         unsigned long           BitVector;
#define WSZ                             (sizeof(BitVector)*8)


#define         WAREBATCH               5000
#define         nthbit(map,n)           map[(n)/WSZ] &
(((BitVector)0x1)<< ((n)%WSZ))
#define         setbit(map,n)           map[(n)/WSZ] |=
(((BitVector)0x1)<< ((n)%WSZ))


#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;
{


        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;
}


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);
}


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);
}


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();


        RandomPermutation(perm, MAXITEMS);


        for (i_id=1;   i_id <= MAXITEMS; i_id++)
        {


        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);


        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);


        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);
}


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);
}


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;
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;


        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);



                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,"");
                        }


                bulk_load(bulk_c);
}

end_customer_load()
{
                bulk_close(bulk_c);
}


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;


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];


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_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);
}




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;



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];

                for (i=0; i<(MAXITEMS+(WSZ-1))/WSZ; i++)
                        bmp[i] = (BitVector)0x0000;

                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();

        for (s_i_id=1;   s_i_id <= MAXITEMS; s_i_id++)
        {
                for (w_id=w1; w_id<=w2; w_id++)
                {

                        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)


		int argc;
		char **argv;
{
		char ch;


		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);

		}


		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;
		}


		if (argc > 4)
		strcpy(password,argv[4]);

		if (w1 <= 0 || w2 > 10000 || 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);


		strcat(zip, "11111");

}



LastName(num, name)

		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[] =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01
23456789";
        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;


        for (i=1; i<=n; i++)
        perm[i] = i;


        for (i=1; i<=n; i++)
        {
        r = RandomNumber(i, n);
        t = perm[i]; perm[i] = perm[r]; perm[r] = t;
        }
}



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;
}
```

loader.h

```
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED

#include <sybfront.h>
#include <sybdb.h>
#include <time.h>



#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


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


extern      int         msg_handler();
extern      int         err_handler();
extern      int         batch_size;
```

```
#endif
```

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
sp_bindcache "c_tinyhot", "tpcc", "sysindexes"
go
sp_bindcache "c_tinyhot", "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
*/

sp_bindcache "c_no", "tpcc", "new_order"
go
sp_bindcache "c_no", "tpcc", "new_order","no_clu"
go

/*
** Cache c_ol
*/

sp_bindcache "c_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_non_customer_index", "tpcc", "customer","c_non1"
go

EOF
```

tpcc_indexes.sh

```
#!/bin/sh -f
echo 'date' "start building indexes"
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 = 100
        on Scache
go
dbcc tune(indextrips, 100, warehouse)
go

create unique clustered index d_clu
        on district(d_w_id, d_id)
        with fillfactor = 100
        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)
        with fillfactor = 100
        on Scidx
go

checkpoint
go
EOF
echo 'date' "finished building indexes"
```

tpcc_tables.sh

```
#!/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 Scust
go
create unique clustered index c_clu
        on customer(c_w_id, c_id, c_d_id)
        on Scust
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 Sordlne
go
create unique clustered index ol_clu
        on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
        on Sordlne
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
```

Appendix C

scr_util.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1999 BY                                   *
 *  COMPAQ COMPUTER CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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 <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <string.h>
#include <ctype.h>
#include <stdarg.h>
#include <stdtypes.h>

#ifdef _WIN32
# include <nt_lib.h>
#endif

#include <prte.h>

#include <common.h>

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

#define SCR_UTIL_C
#include <scr_util.h>



static print_getnet_debug(char *name, char *ret_string, char *def_string);


int
GetCallingThreadLimit( int *pCallingThreadLimit )
{
  *pCallingThreadLimit = PRTEget_worker_thread_limit( );

  return( ERR_SUCCESS );
}


int
GetConfigValue( char *Name, int *Len, char *Value )
{
  int                                   Status;

  Status = GetConfigValueTX( Name, Len, Value );
  if( SCR_E_SUCCESS == Status )
  {

    Status = ERR_SUCCESS;
  }

  return( Status );
}



int
GetConfigValueTX( char *Name, int *Len, char *Value )
{
  char                                  *locValue;
  int                                   locLen;
  char
                FullName[PRTE_MAX_NET_VAR_NAME_LENGTH+1];

  strcpy( FullName, Sut_name );
  strcat( FullName, "_" );
  strcat( FullName, Name );
  locValue = PRTEget_network_variable( FullName );

  if( NULL == locValue )
  {
    Value[0] = '\0';
    *Len = 0;
    return( ERR_CANT_FIND_VALUE );
  }
  else
  {
    locLen = strlen( locValue );
    if( locLen < *Len )
    {
      memcpy( Value, locValue, locLen+1 );
      *Len = locLen;
      return( SCR_E_SUCCESS );
    }
    else
    {
      return( ERR_VALUE_TOO_LONG );
    }
  }
}

int32
GetConnectInfo( int32 RunMode, int32 *pTpccUsers, int32 *pNumFEs,
                              int32 *pAdminUsers,
                              int32 *pNumConnects, connect_t **ppConnects )
{
#define COUNT_UNSPECIFIED -1
#define CONNECT_INFO                  "CONNECT%d"
  char                                  Field[11];
  int                                   Index;
  char                                  NetVarName[50];
  int32                                 NetVarNum;
  int                                   NumCountUnspec;
  char                                  *pConnect;
  char                                  Script[PATH_MAX];
  char                                  *pTmpChar;
  int                                   Remainder;
  int                                   UcntSpec;
  int                                   TotalUcnt;
  int                                   TotalDcnt;
  int                                   Ucnt;


  if( NULL == ( pTmpChar = PRTEget_network_variable(
"USER_SCRIPT_NAME" )))
  {
    Script[0] = '\0';
  }
```

```
  else
  {
   strcpy( Script, pTmpChar );
  }


  if( NULL == ( pTmpChar = PRTEget_network_variable(
"TOTAL_DISTRICT_COUNT" )))
  {
   return( SCRerr( SCR_E_TOTAL_DISTRICT_COUNT_NOT_SET ));
  }
  else
  {
   TotalDcnt = atoi( pTmpChar );
  }


  if( NULL == ( pTmpChar = PRTEget_network_variable(
"TOTAL_USER_COUNT" )))
  {
   TotalUcnt = COUNT_UNSPECIFIED;
  }
  else
  {
   TotalUcnt = atoi( pTmpChar );
  }


  *pNumConnects = 0;
  NetVarNum = 1;
  while( TRUE )
  {
   sprintf( NetVarName, CONNECT_INFO, NetVarNum++ );
   if( NULL == (pConnect = PRTEget_network_variable( NetVarName )))
   {
    break;
   }

   if( '\0' == pConnect[0] )
   {
    continue;
   }
   (*pNumConnects)++;
  }
  if( 0 == *pNumConnects )
  {
   return( SCRerr( SCR_E_NO_CONNECTS ));
  }


  *ppConnects = ( connect_t * ) malloc( *pNumConnects * sizeof(
**ppConnects ));
  if( NULL == *ppConnects )
  {
   return( SCRerr( SCR_E_MALLOC_CONNECT_ARRAY ));
  }


  *pNumFEs = 0;
  *pAdminUsers = 0;
  NumCountUnspec = 0;
  UcntSpec = 0;
  Index = 0;
  NetVarNum = 1;
  while( TRUE )
  {
   sprintf( NetVarName, CONNECT_INFO, NetVarNum++ );
   if( NULL == (pConnect = PRTEget_network_variable( NetVarName )))
   {
    break;
   }

   if( '\0' == pConnect[0] )
   {
    continue;
   }


   pTmpChar = SCR_next_list_element( pConnect,
(*ppConnects)[Index].RTEName );
   if( '\0' == (*ppConnects)[Index].RTEName[0] )
   {
    free( *ppConnects );
    SCRlog( "Parsing %s.", NetVarName );
    return( SCRerr( SCR_E_NO_RTE_IN_CONNECT ));
   }

   pTmpChar = SCR_next_list_element( pTmpChar,
(*ppConnects)[Index].FEName );

   pTmpChar = SCR_next_list_element( pTmpChar, Field );
   if( (*ppConnects)[Index].ConfConn = ( 'T' == Field[0] || 't' == Field[0] ))
   {
    (*pNumFEs)++;
   }

   pTmpChar = SCR_next_list_element( pTmpChar, Field );
   if( (*ppConnects)[Index].AdminConn = ( 'T' == Field[0] || 't' == Field[0]
))
   {
    (*pAdminUsers)++;
   }

   pTmpChar = SCR_next_list_element( pTmpChar, Field );
   if( '\0' == Field[0] )
   {
    (*ppConnects)[Index].Ucnt = COUNT_UNSPECIFIED;
    NumCountUnspec++;
   }
   else
   {
    UcntSpec += (*ppConnects)[Index].Ucnt = atoi( Field );
   }

   pTmpChar = SCR_next_list_element( pTmpChar,
(*ppConnects)[Index].Script );
   if( '\0' == (*ppConnects)[Index].Script[0] )
   {
    if( '\0' == Script[0] )
    {
                free( *ppConnects );
                SCRlog( "Parsing %s.", NetVarName );
                return( SCRerr( SCR_E_USER_SCRIPT_NAME_NOT_SET
));
    }
    strcpy( (*ppConnects)[Index].Script, Script );
   }
   Index++;
  }


  if( 0 == NumCountUnspec )
  {
   if( COUNT_UNSPECIFIED != TotalUcnt && UcntSpec != TotalUcnt )
   {
    free( *ppConnects );
    return( SCRerr( SCR_E_USER_COUNT_MISMATCH ));
   }
   *pTpccUsers = UcntSpec;
  }
  else
  {
   if( COUNT_UNSPECIFIED == TotalUcnt )
   {
    TotalUcnt = TotalDcnt;
   }

   *pTpccUsers = 0;
   Ucnt = ( TotalUcnt - UcntSpec ) / NumCountUnspec;
   Remainder = ( TotalUcnt - UcntSpec ) % NumCountUnspec;
   for( Index = 0; Index < *pNumConnects; Index++ )
   {
    if( COUNT_UNSPECIFIED == (*ppConnects)[Index].Ucnt )
    {
                (*ppConnects)[Index].Ucnt = Ucnt + ( Remainder-- > 0 ? 1 : 0
);
    }
    *pTpccUsers += (*ppConnects)[Index].Ucnt;
   }
  }
```

```
  if( END_TO_END_MODE == RunMode &&
      TotalDcnt != *pTpccUsers )
  {
    free( *ppConnects );
    return( SCRerr( SCR_E_DISTRICT_USER_COUNT_MISMATCH ));
  }

# pragma message ("FIXME: GetConnectInfo: Should warn tester if user
count won't fully access all districts for the last warehouse.")
# pragma message ("FIXME: GetConnectInfo: Should warn tester if user
count on a given network card exceeds limit (currently 1023).")

  return( SCR_E_SUCCESS );
}


int32
SCRerr( int32 ErrCode )
{
  char                                  ErrFmt[] = "\007ERROR( %d ):
%s\n\n";
  int32                                 Idx;
  char                                  *ErrString;

  Idx = 0;
  while( '\0' != ScrErrMsgs[Idx].ErrMsg[0] &&
                 ErrCode != ScrErrMsgs[Idx].ErrCode )
  {
    Idx++;
  }
  if( '\0' != ScrErrMsgs[Idx].ErrMsg[0] )
  {
    ErrString = ScrErrMsgs[Idx].ErrMsg;
  }
  else
  {

    Idx = 0;
    while( '\0' != errorMsgs[Idx].szMsg[0] &&
                   ErrCode != errorMsgs[Idx].iError )
    {
      Idx++;
    }
    if( '\0' != errorMsgs[Idx].szMsg[0] )
    {
      ErrString = errorMsgs[Idx].szMsg;
    }
    else
    {
      ErrString = ScrErrMsgs[SCR_E_NO_ERR_MSG -
SCR_E_BASE].ErrMsg;
    }
  }

  SCRlog( ErrFmt, ErrCode, ErrString );
  return( ErrCode );
}


void
SCRlog( char *Format, ... )
{
  char                          Buf[8192];
  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();
    if( 0.0 == Randval )
    {


      continue;
    }
    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];

  SCRlog( "Netvar: name = %s netvar = %s, default = %s\n",
                name, ret_string, def_string );
}
#endif


char
*SCR_next_list_element( char *List, char *Element )
{
#define EAT_LEADING_WHITESPACE(p) while( isspace(*(p)) ) (p)++
#define EAT_TRAILING_WHITESPACE(p) while( isspace(*((p)-1)) ) (p)--
#define IS_SEPARATOR(c) (( ',' == (c)))
  char *pTmpChar;

  pTmpChar = Element;
  if( NULL != pTmpChar )
  {
    *pTmpChar = '\0';
  }

  if( List && *List )
  {
    EAT_LEADING_WHITESPACE( List );
    while( '\0' != *List && !IS_SEPARATOR( *List ))
    {
      if( '\\' == *List )
      {
                List++;
                if( '\0' == *List )
                {
                  break;
                }
      }
      if( NULL != pTmpChar )
      {
                *pTmpChar++ = *List;
      }
      List++;
    }
    if( NULL != pTmpChar )
    {
      EAT_TRAILING_WHITESPACE( pTmpChar );
      *pTmpChar = '\0';
    }


    if( IS_SEPARATOR( *List ))
    {
      List++;
    }

    return( List );
  }
  else
  {
    return( NULL );
  }
}


int
SCR_count_elements( char *List )
{
  char *pTmpChar;
  int Count = 0;


  pTmpChar = List;
  while( NULL != ( pTmpChar = SCR_next_list_element( pTmpChar,
NULL )))
  {
    Count++;
  }



  if( 0 == Count )
  {
    Count++;
  }

  return Count;
}


void
SCR_make_delta_time_string( char *timestr, double DeltaSeconds )
{
#define ONE_MINUTE 60
```

```
#define ONE_HOUR (60*ONE_MINUTE)
  int                                   Hours;
  int                                   Minutes;
  int                                   Seconds;
  int                                   MilliSeconds;

  Seconds = (int)DeltaSeconds;
  MilliSeconds = (int)( DeltaSeconds*1000.0 - Seconds*1000.0 );

  Hours = Seconds / ONE_HOUR;
  Seconds = Seconds % ONE_HOUR;

  Minutes = Seconds / ONE_MINUTE;
  Seconds = Seconds % ONE_MINUTE;

  sprintf(timestr,"%2d:%02d:%02d.%03d", Hours, Minutes, Seconds,
MilliSeconds);
}

int
SetConfigValueTX( char *Name, char *Value )
{
  char
              FullName[PRTE_MAX_NET_VAR_NAME_LENGTH+1];

  strcpy( FullName, Sut_name );
  strcat( FullName, "_" );
  strcat( FullName, Name );

  PRTEset_network_variable( FullName, Value );

  return( SCR_E_SUCCESS );
}
```

scr_util.h

```
#ifndef SCR_UTIL_H
#define SCR_UTIL_H
/*+***********************************************************
**************
 *                                                           *
 *  COPYRIGHT (c) 1999 BY                                    *
 *  DIGITAL COMPUTER CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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
DIGITAL COMPUTER   *
 *  CORPORATION.                                             *
 *                                                           *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                           *
 *                                                           *

***********************************************************
*************-*/

#define ATOI(pStr,pNext,val)\
  {\
    int                                 ival;\
    char                        *pSrc = (pStr);\
    ival = 0;\
    while ('0' <= *pSrc && *pSrc <= '9')\
    {\
      ival = (ival * 10) + (*pSrc - '0');\
      pSrc++;\
    }\
    (pNext) = pSrc;\
    (val) = ival;\
  }

#define ITOA(val,pStr,pNext)\
  {\
    int                                         ival;\
    char                        *pDst = (pStr);\
    char                        Tmp[12]; \
    char                        *pTmp = &Tmp[0];\
    ival = (val);\
    if( 0 > ival )\
    {\
      *pDst++ = '-';\
      ival = -ival;\
    }\
    *pTmp++ = '\0';\
    if( 0 == ival )\
    {\
       *pTmp++ = '0';\
    }\
    while( 0 < ival )\
    {\
      *pTmp++ = ( ival % 10 ) + '0';\
      ival /= 10;\
    }\
    while( '\0' != ( *pDst++ = *--pTmp ));\
    (pNext) = pDst;\
  }


typedef struct _connect_t
{
  BOOL                                  ConfConn;
  BOOL                                  ConfMod;
  BOOL                                  AdminConn;
  int32                                 AideID;
  char                                  RTEName[MAXHOSTNAMELEN];
  char                                  FEName[MAXHOSTNAMELEN];
  int32                                 Ucnt;
  char                                  Script[PATH_MAX];
} connect_t;


union dataptr {
  double *pDouble;
  int *pInt;
  float *pFloat;
};


int  GetCallingThreadLimit( int *pCallingThreadLimit );
int32 GetConnectInfo( int32 RunMode, int32 *pTpccUsers, int32
*pNumFEs,
                                  int32 *pAdminUsers,
                                  int32 *pNumConnects, connect_t
**ppConnects );

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);
void SCR_make_delta_time_string( char *timestr, double DeltaSeconds );
int SetConfigValueTX( char *Name, char *Value );
int GetConfigValueTX( char *Name, int *Len, char *Value );
```

```
#define     SCR_E_SUCCESS
            (0)

#define SCR_E_BASE
            200000

#define SCR_E_NO_ERR_MSG
            (SCR_E_BASE+1)
#define SCR_E_INIT
            (SCR_E_BASE+2)
#define SCR_E_MAKING_RUN_DIR
            (SCR_E_BASE+3)
#define SCR_E_NO_REDUCER
            (SCR_E_BASE+4)
#define SCR_E_CKPT_INT_TOO_LONG
            (SCR_E_BASE+5)
#define SCR_E_CKPT_INT_GREATER_THAN_MEASUREMENT_INT
            (SCR_E_BASE+6)
#define SCR_E_MEASUREMENT_NOT_INTEGRAL_MULTIPLE
            (SCR_E_BASE+7)
#define SCR_E_WARMUP_TOO_SHORT
            (SCR_E_BASE+8)
#define SCR_E_CANT_PARSE_SERVICE_CONTROL_PAGE
            (SCR_E_BASE+9)
#define SCR_E_CANT_STOP_WEB
            (SCR_E_BASE+10)
#define SCR_E_CANT_START_WEB
            (SCR_E_BASE+11)
#define SCR_E_AIDE_TIMEOUT_STOP_FE
            (SCR_E_BASE+12)
#define SCR_E_GET_BE_INFO
            (SCR_E_BASE+13)
#define SCR_E_INVALID_RUN_MODE
            (SCR_E_BASE+14)
#define SCR_E_AUDIT_FUNC_NETVARS
            (SCR_E_BASE+15)
#define SCR_E_BE_NAMES
            (SCR_E_BASE+16)
#define SCR_E_MALLOC_BE_ARRAY
            (SCR_E_BASE+17)
#define SCR_E_CODE_VERSION_NOT_SET
            (SCR_E_BASE+18)
#define SCR_E_RUN_DIR_NOT_SET
            (SCR_E_BASE+19)
#define SCR_E_CODE_VERSION_MISMATCH
            (SCR_E_BASE+20)
#define SCR_E_OPEN_BIN_LOG_FILE
            (SCR_E_BASE+21)
#define SCR_E_WRITE_BIN_LOG_FILE_HEADER_SIZE
            (SCR_E_BASE+22)
#define SCR_E_WRITE_BIN_LOG_FILE_HEADER
            (SCR_E_BASE+23)
#define SCR_E_MASTER_ID_NOT_SET
            (SCR_E_BASE+24)
#define SCR_E_RUN_NUMBER_NOT_SET
            (SCR_E_BASE+25)
#define SCR_E_VERSION_NUMBER_NOT_SET
            (SCR_E_BASE+26)
#define SCR_E_REDUCER_UPDATE_INTERVAL_NOT_SET
            (SCR_E_BASE+27)
#define SCR_E_REDUCER_HEADER_INTERVAL_NOT_SET
            (SCR_E_BASE+28)
#define SCR_E_OPEN_AIDE_DATA_FILE
            (SCR_E_BASE+29)
#define SCR_E_DATABASE_TYPE_NOT_SET
            (SCR_E_BASE+30)
#define SCR_E_INVALID_DATABASE_TYPE
            (SCR_E_BASE+31)
#define SCR_E_AIDE_INIT_TIMEOUT
            (SCR_E_BASE+32)
#define SCR_E_TOTAL_DISTRICT_COUNT_NOT_SET
            (SCR_E_BASE+33)
#define SCR_E_NO_CONNECTS
            (SCR_E_BASE+34)
#define SCR_E_NO_RTE_IN_CONNECT
            (SCR_E_BASE+35)
#define SCR_E_CALC_VALID_TXN_MIX
            (SCR_E_BASE+36)
#define SCR_E_REDUCER_INIT_TIMEOUT
            (SCR_E_BASE+37)
#define SCR_E_LOAD_FE_TIMEOUT
            (SCR_E_BASE+38)
#define SCR_E_CONNECT_APPLICATION
            (SCR_E_BASE+39)
#define SCR_E_SEND
            (SCR_E_BASE+40)
#define SCR_E_WAIT_FOR
            (SCR_E_BASE+41)
#define SCR_E_HTTP_STATUS_OFFSET
            (SCR_E_BASE+42)
#define SCR_E_FORM_TYPE_OFFSET
            (SCR_E_BASE+43)
#define SCR_E_WEB_PAGE_TOO_SHORT
            (SCR_E_BASE+44)
#define SCR_E_WEB_PAGE_STATUS_CODE
            (SCR_E_BASE+45)
#define SCR_E_ERROR_STATUS_CODE
            (SCR_E_BASE+46)
#define SCR_E_ERROR_WEB_PAGE
            (SCR_E_BASE+47)
#define SCR_E_WRONG_WEB_PAGE
            (SCR_E_BASE+48)
#define SCR_E_AIDE_TASK_TIMEOUT
            (SCR_E_BASE+49)
#define SCR_E_AIDES_FAILED_TASK
            (SCR_E_BASE+50)
#define SCR_E_BE_CKPT_TIMEOUT
            (SCR_E_BASE+51)
#define SCR_E_BE_CKPT_FAILED
            (SCR_E_BASE+52)
#define SCR_E_AIDE_CKPT_INIT_TIMEOUT
            (SCR_E_BASE+53)
#define SCR_E_USER_LOGIN_TIMEOUT
            (SCR_E_BASE+54)
#define SCR_E_USER_LOGIN_FAILED
            (SCR_E_BASE+55)
#define SCR_E_LOG_DIR_NOT_SET
            (SCR_E_BASE+56)
#define SCR_E_CGI_SCRIPT_NAME_NOT_SET
            (SCR_E_BASE+57)
#define SCR_E_INVALID_TXN_TYPE_REQUESTED
            (SCR_E_BASE+58)
#define SCR_E_USER_START_FAILED
            (SCR_E_BASE+59)
#define SCR_E_USER_START_TIMEOUT
            (SCR_E_BASE+60)
#define SCR_E_USER_STOP_FAILED
            (SCR_E_BASE+61)
#define SCR_E_USER_STOP_TIMEOUT
            (SCR_E_BASE+62)
#define SCR_E_AIDE_TIMEOUT_START_FE
            (SCR_E_BASE+63)
#define SCR_E_AIDES_FAILED_START
            (SCR_E_BASE+64)
#define SCR_E_UNUSED_4
            (SCR_E_BASE+65)
#define SCR_E_UNUSED_5
            (SCR_E_BASE+66)
#define SCR_E_AIDE_CKPT_RUN_TIMEOUT
            (SCR_E_BASE+67)
#define SCR_E_AIDE_CKPT_STOP_TIMEOUT
            (SCR_E_BASE+68)
#define SCR_E_WRITE_SENDERS_ID
            (SCR_E_BASE+69)
#define SCR_E_WRITE_MSG_TYPE
            (SCR_E_BASE+70)
#define SCR_E_WRITE_MSG_SIZE
            (SCR_E_BASE+71)
#define SCR_E_WRITE_MSG
            (SCR_E_BASE+72)
#define SCR_E_AIDE_EXIT_TIMEOUT
            (SCR_E_BASE+73)
#define SCR_E_OPEN_C_LAST_FILE
            (SCR_E_BASE+74)
#define SCR_E_SYNC_USER_TIMEOUT
            (SCR_E_BASE+75)
#define SCR_E_MALLOC_CONNECT_ARRAY
            (SCR_E_BASE+76)
#define SCR_E_MALLOC_USER_STOP_ARRAY
            (SCR_E_BASE+77)
```

```
#define SCR_E_AIDES_FAILED_STOP
                (SCR_E_BASE+78)
#define SCR_E_ACTION_NOT_SET
                (SCR_E_BASE+79)
#define SCR_E_UNUSED_1
                (SCR_E_BASE+80)
#define SCR_E_DELI_OFFSET
                (SCR_E_BASE+81)
#define SCR_E_DURA_PARSE_WAREHOUSE
                (SCR_E_BASE+82)
#define SCR_E_DURA_PARSE_DISTRICT
                (SCR_E_BASE+83)
#define SCR_E_DURA_PARSE_ORDER_ID
                (SCR_E_BASE+84)
#define SCR_E_GREETING
                (SCR_E_BASE+85)
#define SCR_E_CONNECT_ABORTED
                (SCR_E_BASE+86)
#define SCR_E_MENU_BAR
                (SCR_E_BASE+87)
#define SCR_E_AIDE_GET_LOG_SIZE
                (SCR_E_BASE+88)
#define SCR_E_RUN_ID_NOT_SET
                (SCR_E_BASE+89)
#define SCR_E_AIDE_SKIPPING_STATS
                (SCR_E_BASE+90)
#define SCR_E_AIDE_SWITCH_LOG
                (SCR_E_BASE+91)
#define SCR_E_NO_VALUE
                (SCR_E_BASE+92)
#define SCR_E_NO_TYPE
                (SCR_E_BASE+93)
#define SCR_E_NO_DATA
                (SCR_E_BASE+94)
#define SCR_E_UNUSED_6
                (SCR_E_BASE+95)
#define SCR_E_REDUCER_ID_NOT_SET
                (SCR_E_BASE+96)
#define SCR_E_RTE_NAMES_NOT_SET
                (SCR_E_BASE+97)
#define SCR_E_USER_COUNT_MISMATCH
                (SCR_E_BASE+98)
#define SCR_E_LOAD_FE_FAILURE
                (SCR_E_BASE+99)
#define SCR_E_USER_CREATE_FAILED
                (SCR_E_BASE+100)
#define SCR_E_USER_INIT_FAILED
                (SCR_E_BASE+101)
#define SCR_E_USER_INIT_TIMEOUT
                (SCR_E_BASE+102)
#define SCR_E_FIRST_USER_ID_NOT_SET
                (SCR_E_BASE+103)
#define SCR_E_VALUE_MODIFIED
                (SCR_E_BASE+104)
#define SCR_E_CANT_PARSE_GET_REG_PAGE
                (SCR_E_BASE+105)
#define SCR_E_CANT_INIT_TIME_COUNTER
                (SCR_E_BASE+106)
#define SCR_E_USER_SCRIPT_NAME_NOT_SET
                (SCR_E_BASE+107)
#define SCR_E_WRONG_DATA_ITEM_COUNT
                (SCR_E_BASE+108)
#define SCR_E_AIDE_TIMEOUT_CONFIG_FE
                (SCR_E_BASE+109)
#define SCR_E_AIDES_FAILED_CONFIG
                (SCR_E_BASE+110)
#define SCR_E_NOT_CORRECT_TRANS
                (SCR_E_BASE+111)
#define SCR_E_INVALID_CONNECT
                (SCR_E_BASE+112)
#define SCR_E_INVALID_DISCONNECT
                (SCR_E_BASE+113)
#define SCR_E_UI_NOT_SET
                (SCR_E_BASE+114)
#define SCR_E_UI_LOAD_FAILED
                (SCR_E_BASE+115)
#define SCR_E_NO_PARSE_INIT_FAILED
                (SCR_E_BASE+116)
#define SCR_E_DURA_PARSE_EXECUTION_STATUS
                (SCR_E_BASE+117)
#define SCR_E_UI_UNLOAD_FAILED
                (SCR_E_BASE+118)
#define SCR_E_AIDE_CKPT_DISCONNECT_TIMEOUT
                (SCR_E_BASE+119)
#define SCR_E_DISTRICT_USER_COUNT_MISMATCH
                (SCR_E_BASE+120)
#define SCR_E_REDUCER_CREATE_FAILED
                (SCR_E_BASE+121)
#define SCR_E_AIDE_CREATE_FAILED
                (SCR_E_BASE+122)
#define SCR_E_CUSTOMER_INQUIRY_PARSE_C_ID
                (SCR_E_BASE+123)
#define SCR_E_PRIME_IIS
                (SCR_E_BASE+124)

#define SCR_E_MAX_ERROR
                (SCR_E_BASE+124)


typedef struct _scr_e_msg_t
{
  int32                 ErrCode;
  char                  ErrMsg[256];
} scr_e_msg_t;

#ifdef SCR_UTIL_C

scr_e_msg_t ScrErrMsgs[] =
{
  { SCR_E_SUCCESS, "Success, no error." },
  { SCR_E_NO_ERR_MSG, "No error message available for this error
code." },
  { SCR_E_INIT, "Failed to initialize." },
  { SCR_E_MAKING_RUN_DIR, "Failed to make the run directory." },
  { SCR_E_NO_REDUCER, "Failed to get the reducer's PRTE user id." },
  { SCR_E_CKPT_INT_TOO_LONG, "Checkpoint interval > 1800 seconds,
this violates the benchmark specification." },
  { SCR_E_CKPT_INT_GREATER_THAN_MEASUREMENT_INT,
"Checkpoint interval > measurement interval, this violates the benchmark
specification." },
  { SCR_E_MEASUREMENT_NOT_INTEGRAL_MULTIPLE,
"Measurement interval is not an integral multiple of checkpoint interval, this
violates the benchmark specification." },
  { SCR_E_WARMUP_TOO_SHORT, "The warmup interval is too short to
support requested checkpoint interval." },
  { SCR_E_CANT_PARSE_SERVICE_CONTROL_PAGE, "Error parsing
the service control response page." },
  { SCR_E_CANT_STOP_WEB, "Unable to stop the Web Server." },
  { SCR_E_CANT_START_WEB, "Unable to start the Web Server." },
  { SCR_E_AIDE_TIMEOUT_STOP_FE, "Timed out waiting for aides to
stop application code." },
  { SCR_E_GET_BE_INFO, "Problem getting back end information." },
  { SCR_E_INVALID_RUN_MODE, "Invalid run mode specified." },
  { SCR_E_AUDIT_FUNC_NETVARS, "Problem getting audit function
network variables." },
  { SCR_E_BE_NAMES, "Unable to get BE_NAMES network variable." },
  { SCR_E_MALLOC_BE_ARRAY, "Unable to malloc space for BE
array." },
  { SCR_E_CODE_VERSION_NOT_SET, "Code version network variable
not set." },
  { SCR_E_RUN_DIR_NOT_SET, "Run directory network variable not
set." },
  { SCR_E_CODE_VERSION_MISMATCH, "Code version mismatch with
Master." },
  { SCR_E_OPEN_BIN_LOG_FILE, "Failed to open the binary data log
file." },
  { SCR_E_WRITE_BIN_LOG_FILE_HEADER_SIZE, "Error writing
binary data log file header size." },
  { SCR_E_WRITE_BIN_LOG_FILE_HEADER, "Error writing binary data
log file header." },
  { SCR_E_MASTER_ID_NOT_SET, "Master ID network variable not set."
},
  { SCR_E_RUN_NUMBER_NOT_SET, "Run number network variable
not set." },
  { SCR_E_VERSION_NUMBER_NOT_SET, "Version number network
variable not set." },
  { SCR_E_REDUCER_UPDATE_INTERVAL_NOT_SET, "Reducer
update interval network variable not set." },
  { SCR_E_REDUCER_HEADER_INTERVAL_NOT_SET, "Reducer
header update interval network variable not set." },
```

```
  { SCR_E_OPEN_AIDE_DATA_FILE, "Error opening aide data file." },
  { SCR_E_DATABASE_TYPE_NOT_SET, "Database type network
variable not set." },
  { SCR_E_INVALID_DATABASE_TYPE, "Invalid database type" },
  { SCR_E_AIDE_INIT_TIMEOUT, "Timed out waiting for Aides to
initialize." },
  { SCR_E_TOTAL_DISTRICT_COUNT_NOT_SET,
"TOTAL_DISTRICT_COUNT network variable not set. " },
  { SCR_E_NO_CONNECTS, "No connect network variables have been
specified." },
  { SCR_E_NO_RTE_IN_CONNECT, "No RTE was specified in a
CONNECT network variable." },
  { SCR_E_CALC_VALID_TXN_MIX, "Failed to calculate a valid
transaction mix." },
  { SCR_E_REDUCER_INIT_TIMEOUT, "Timed out waiting for Reducer
to initialize." },
  { SCR_E_LOAD_FE_TIMEOUT, "Timed out waiting for FE(s) to load."
},
  { SCR_E_CONNECT_APPLICATION, "Error trying to connect to the
application (SUT)." },
  { SCR_E_SEND, "Error sending data to the application (SUT)." },
  { SCR_E_WAIT_FOR, "Error waiting for reply from application (SUT)."
},
  { SCR_E_HTTP_STATUS_OFFSET, "Couldn't determine HTTP status
offset." },
  { SCR_E_FORM_TYPE_OFFSET, "Couldn't determine the form type
offset." },
  { SCR_E_WEB_PAGE_TOO_SHORT, "Received a web page that is too
short to be from our dll." },
  { SCR_E_WEB_PAGE_STATUS_CODE, "Couldn't find form status code
in web page." },
  { SCR_E_ERROR_STATUS_CODE, "Received an error page from our
dll, but couldn't parse the status code." },
  { SCR_E_ERROR_WEB_PAGE, "Received and error page from our dll."
},
  { SCR_E_WRONG_WEB_PAGE, "Received a valid web page from our
dll, but it isn't the one we expected." },
  { SCR_E_AIDE_TASK_TIMEOUT, "Timed out waiting for all aides to
perform a task." },
  { SCR_E_AIDES_FAILED_TASK, "One or more aides failed to perform a
task." },
  { SCR_E_BE_CKPT_TIMEOUT, "Timed out waiting for all aides to do
their backend checkpoint init." },
  { SCR_E_BE_CKPT_FAILED, "One or more aides failed to do their
backend checkpoint init." },
  { SCR_E_AIDE_CKPT_INIT_TIMEOUT, "Timed out waiting for all
aides to do their checkpoint init." },
  { SCR_E_USER_LOGIN_TIMEOUT, "Timed out waiting for all users to
confirm login." },
  { SCR_E_USER_LOGIN_FAILED, "A user failed to log in successfully."
},
  { SCR_E_LOG_DIR_NOT_SET, "Log dir network variable not set." },
  { SCR_E_CGI_SCRIPT_NAME_NOT_SET, "CGI script name network
variable not set. " },
  { SCR_E_INVALID_TXN_TYPE_REQUESTED, "An invalid transcation
type was requested." },
  { SCR_E_USER_START_FAILED, "User(s) failed to start doing
transactions." },
  { SCR_E_USER_START_TIMEOUT, "Timed out waiting for all users to
start doing transactions." },
  { SCR_E_USER_STOP_FAILED, "User(s) failed to confirm stop." },
  { SCR_E_USER_STOP_TIMEOUT, "Timed out waiting for all users to
confirm stop." },
  { SCR_E_AIDE_TIMEOUT_START_FE, "Timed out waiting for aides to
start application code." },
  { SCR_E_AIDES_FAILED_START, "Not all aides successfully started
application code." },
  { SCR_E_UNUSED_4, "Unused message 4." },
  { SCR_E_UNUSED_5, "Unused message 5." },
  { SCR_E_AIDE_CKPT_RUN_TIMEOUT, "Timed out waiting for aides to
start checkpoint loop." },
  { SCR_E_AIDE_CKPT_STOP_TIMEOUT, "Timed out waiting for aides
to stop checkpoint loop." },
  { SCR_E_WRITE_SENDERS_ID, "Failed to write sender's id to binary
log file." },
  { SCR_E_WRITE_MSG_TYPE, "Failed to write message type to binary
log file." },
  { SCR_E_WRITE_MSG_SIZE, "Failed to write message size to binary log
file." },
  { SCR_E_WRITE_MSG, "Failed to write message to binary log file." },
  { SCR_E_AIDE_EXIT_TIMEOUT, "Timed out waiting for all aides to
exit." },
  { SCR_E_OPEN_C_LAST_FILE, "Unable to open file to write C_LAST
constant." },
  { SCR_E_SYNC_USER_TIMEOUT, "Timed out on delay to allow users
to catch up with in sync offset." },
  { SCR_E_MALLOC_CONNECT_ARRAY, "Unable to malloc space for
the connect array." },
  { SCR_E_MALLOC_USER_STOP_ARRAY, "Unable to malloc space for
the user stop array." },
  { SCR_E_AIDES_FAILED_STOP, "Not all aides successfully stopped
application code." },
  { SCR_E_ACTION_NOT_SET, "Action network variable not set." },
  { SCR_E_UNUSED_1, "Unused message 1." },
  { SCR_E_DELI_OFFSET, "Couldn't determine deli record offset." },
  { SCR_E_DURA_PARSE_WAREHOUSE, "Couldn't locate warehouse ID
in response page." },
  { SCR_E_DURA_PARSE_DISTRICT, "Couldn't locate district ID in
response page." },
  { SCR_E_DURA_PARSE_ORDER_ID, "Couldn't locate order ID in
response page." },
  { SCR_E_GREETING, "Failed to get the Greeting Form." },
  { SCR_E_CONNECT_ABORTED, "Asked to stop while logging in." },
  { SCR_E_MENU_BAR, "Failed to get the Menu Bar Form." },
  { SCR_E_AIDE_GET_LOG_SIZE, "Failed to get log size." },
  { SCR_E_RUN_ID_NOT_SET, "Run ID network variable not set." },
  { SCR_E_AIDE_SKIPPING_STATS, "WARNING: Unable to read file --
skipping Oracle stats." },
  { SCR_E_AIDE_SWITCH_LOG, "Failed to switch log." },
  { SCR_E_NO_VALUE, "No value found." },
  { SCR_E_NO_TYPE, "No type found." },
  { SCR_E_NO_DATA, "No data found." },
  { SCR_E_UNUSED_6, "Unused message 6." },
  { SCR_E_REDUCER_ID_NOT_SET, "REDUCER_ID network variable
not set." },
  { SCR_E_RTE_NAMES_NOT_SET, "RTE names network variable not
set." },
  { SCR_E_USER_COUNT_MISMATCH, "The count of users in each
CONNECT do not sum to the TOTAL_USER_COUNT." },
  { SCR_E_LOAD_FE_FAILURE, "FE(s) failed to load correctly." },
  { SCR_E_USER_CREATE_FAILED, "The creation of TPC-C users
failed." },
  { SCR_E_USER_INIT_FAILED, "User(s) failed to confirm initialization."
},
  { SCR_E_USER_INIT_TIMEOUT, "Timed out waiting for all users to
confirm initialization." },
  { SCR_E_FIRST_USER_ID_NOT_SET, "FIRST_USER_ID network
variable not set." },
  { SCR_E_VALUE_MODIFIED, "Registry setting was modified." },
  { SCR_E_CANT_PARSE_GET_REG_PAGE, "Error parsing GetRegistry
response page." },
  { SCR_E_CANT_INIT_TIME_COUNTER, "Error initializing time value
used for scale calculation of tpmC." },
  { SCR_E_USER_SCRIPT_NAME_NOT_SET, "User script name not
specified in CONNECT nor is USER_SCRIPT_NAME network variable
set." },
  { SCR_E_WRONG_DATA_ITEM_COUNT, "Configuration value list
must have either one value or match the number of CONNECT#
statements." },
  { SCR_E_AIDE_TIMEOUT_CONFIG_FE, "Timed out waiting for all
aides to configure FEs." },
  { SCR_E_AIDES_FAILED_CONFIG, "Not all aides successfully
configured FEs." },
  { SCR_E_NOT_CORRECT_TRANS, "Did not recieve the expected
transaction response." },
  { SCR_E_INVALID_CONNECT, "The specified connection mode is
invalid." },
  { SCR_E_INVALID_DISCONNECT, "The specified disconnect mode is
invalid." },
  { SCR_E_UI_NOT_SET, "The UI network variable is either not defined or
not properly defined. " },
  { SCR_E_UI_LOAD_FAILED, "UI load failed. " },
  { SCR_E_NO_PARSE_INIT_FAILED, "Unable to initialize NewOrder
parsing function." },
  { SCR_E_DURA_PARSE_EXECUTION_STATUS, "Unable to initialize
durability parsing function." },
  { SCR_E_UI_UNLOAD_FAILED, "UI unload failed. " },
  { SCR_E_AIDE_CKPT_DISCONNECT_TIMEOUT, "Timed out waiting
for all aides to do their checkpoint disconnect." },
```

```
  { SCR_E_DISTRICT_USER_COUNT_MISMATCH, "The specified
district and user count(s) don't agree."},
  { SCR_E_REDUCER_CREATE_FAILED, "Failed to create Reducer." },
  { SCR_E_AIDE_CREATE_FAILED, "Failed to create Aides." },
  { SCR_E_CUSTOMER_INQUIRY_PARSE_C_ID, "Failed to find
customer ID in Customer-Inquiry response." },
  { SCR_E_PRIME_IIS, "Failed to prime the IIS server." },
  { 0, "" }
};
#else
extern scr_e_msg_t ScrErrMsgs[];
#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 Digital Equipment 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              Digital
Equipment Corp.
 *                                Bill Carr      Digital Equipment Corp.
 *
 */
/*+*************************************************************
***************
 *                                                    *
 * COPYRIGHT (c) 1997, 2000 BY                        *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.              *
 * 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
DIGITAL EQUIPMENT   *
 * CORPORATION.                                       *
 *                                                    *
 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR
RELIABILITY OF ITS   *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.             *
 *                                                    *
 *                                                    *

*************************************************************
*************-*/


#ifdef _WIN32
# include <windows.h>
# include <sys\timeb.h>
# include <io.h>
#else
char *
_strupr( char *str )
{
 char *p = str;
 while( '\0' != *p ) { *p = toupper( *p ); p++; }
 return( str );
}
#endif

#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.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;
 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;

 pNewOrderData->o_all_local = 1;


 for(i=0, items=0; i<MAX_OL; i++)
```

```
 {
  if( !GetValuePtr(pProcessedQuery, i*3+IID00, &ptr))
   return ERR_NEWORDER_MISSING_IID_KEY;
  if(*ptr != '&' && *ptr)
  {
   if(!GetNumeric(ptr, &pNewOrderData->o_ol[items].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[items].ol_supply_w_id))
            return ERR_NEWORDER_SUPPW_INVALID;
   if ( pNewOrderData->o_all_local &&
             pNewOrderData->o_ol[items].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[items].ol_quantity))
            return ERR_NEWORDER_QTY_INVALID;
   if ( pNewOrderData->o_ol[items].ol_i_id >= 1000000 ||
             pNewOrderData->o_ol[items].ol_i_id < 1 )
            return ERR_NEWORDER_ITEMID_RANGE;
   if ( pNewOrderData->o_ol[items].ol_quantity >= 100 ||
             pNewOrderData->o_ol[items].ol_quantity < 1 )
            return ERR_NEWORDER_QTY_RANGE;
   items++;
  }
  else
  {
   if(!GetValuePtr(pProcessedQuery, i*3+SP00, &ptr))
            return ERR_NEWORDER_MISSING_SUPPW_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;
  }
 }
 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 = INVALID_C_ID;
  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;


 if(*ptr == '&' || !(*ptr))
 {
  pPaymentData->c_id = INVALID_C_ID;
  pSzTmp = szTmp;
  if ( !GetValuePtr(pProcessedQuery, CLT_P, &ptr) )
   return ERR_PAYMENT_MISSING_CLT;
  if (*ptr == '&' || !(*ptr))
   return ERR_PAYMENT_MISSING_CID_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 (!GetNumeric(ptr, &pPaymentData->c_id))
    return ERR_PAYMENT_CUSTOMER_INVALID;
  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;
}
```

tpcc.h

```
#ifndef TPCC_H
#define TPCC_H

/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1997, 2000 BY                             *
 *  DIGITAL EQUIPMENT CORPORATION, MAYNARD,
MASSACHUSETTS.              *
 *  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
DIGITAL EQUIPMENT   *
 *  CORPORATION.                                            *
 *                                                          *
 *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR
RELIABILITY OF ITS   *
 *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY
DIGITAL.              *
 *                                                          *
 *                                                          *
**********************************************************
*************-*/

/*+
 * 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:
 *
 *
 */


#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

#endif
```

tpcc_gen.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1999 BY                                   *
 *  COMPAQ COMPUTER CORPORATION, MAYNARD,
MASSACHUSETTS.              *
 *  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 <stdio.h>
#include <math.h>
#include <string.h>
#include <time.h>
#include <stdtypes.h>

#ifdef _WIN32
# include <nt_lib.h>
#else
```

```
# include <sys/param.h>
#endif

#include <common.h>

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

#include <scr_util.h>

#include <msg.h>
#include <tx_api.h>
#include <tpcc_gen.h>


#define C_ID_CONST 511
#define ITEM_CONST 4093



#define PCT_C_LAST 60


#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_wdid( txn_context_t *pContext, int *pw_id, int *pd_id )
{
  if( BACKEND_MODE == pContext->RunMode )
  {
    *pw_id = random_int( 1, pContext->max_w_id );
    *pd_id = random_int( 1, DISTRICTS_PER_WAREHOUSE );
  }
  else
  {
    *pw_id = pContext->w_id;
    *pd_id = pContext->d_id;
  }
}

void
get_no_data( txn_context_t *pContext, pNewOrderData pNewOrder,
                    int *context )
{
  int remote_txn = 0;
  int line;

  pNewOrder->CC = (CallersContext)pContext;

  *context = NEW_ORDER_TXN;

  get_wdid( pContext, &pNewOrder->w_id, &pNewOrder->ld_id );

  pNewOrder->d_id = random_int( 1, DISTRICTS_PER_WAREHOUSE );

  pNewOrder->c_id = NURand( 1023, 1, MAX_C_ID, C_ID_CONST );

  pNewOrder->o_ol_cnt = random_int( MIN_OL, MAX_OL );
  for( line = 0; line < pNewOrder->o_ol_cnt; line++ )
  {

    if(( random_int( 0, 999 ) < 990) || ( pContext->max_w_id == 1 ))
    {
      pNewOrder->o_ol[line].ol_supply_w_id = pNewOrder->w_id;
    }
    else
    {
      remote_txn++;
      pNewOrder->o_ol[line].ol_supply_w_id =
                random_int( 1, pContext->max_w_id-1 );
      if( pNewOrder->o_ol[line].ol_supply_w_id >= pNewOrder->w_id )
      {
                pNewOrder->o_ol[line].ol_supply_w_id++;
      }
    }


    if(( line == pNewOrder->o_ol_cnt-1 ) &&
      ( random_int( 0, 999 ) < 10 ))
    {
      pNewOrder->o_ol[line].ol_i_id = ILLEGAL_ITEM;
      *context |= ILLEGAL_ITEM_TXN;
    }
    else
    {
      pNewOrder->o_ol[line].ol_i_id =
                NURand( 8191, 1, MAX_I_ID_POPULATED, ITEM_CONST
);
    }

    pNewOrder->o_ol[line].ol_quantity = random_int( 1,
MAX_OL_QUANTITY );

  }

  *context |= ( pNewOrder->o_ol_cnt << OL_CNT_BIT_POS );
  *context |= ( remote_txn << REMOTE_TXN_BIT_POS );

}

void
get_pt_data( txn_context_t *pContext, pPaymentData pPayment,
                    int *context )
{
  char                                  *ptr;
  int                                   index, i1, i2, i3;

  pPayment->CC = (CallersContext)pContext;

  *context = PAYMENT_TXN;

  get_wdid( pContext, &pPayment->w_id, &pPayment->ld_id );

  pPayment->d_id = random_int( 1, DISTRICTS_PER_WAREHOUSE );

  pPayment->c_id = INVALID_C_ID;

  ptr = pPayment->c_last;
  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
  {

    pPayment->c_id = NURand( 1023, 1, MAX_C_ID, C_ID_CONST );
  }


  if(( random_int( 0,999 ) < 850 ) || ( pContext->max_w_id == 1 ))
  {
    pPayment->c_w_id = pPayment->w_id;
    pPayment->c_d_id = pPayment->ld_id;
  }
  else
  {
    *context |= REMOTE_PAYMENT;
    pPayment->c_w_id = random_int( 1, pContext->max_w_id-1 );
    if( pPayment->c_w_id >= pPayment->w_id )
```

```
    {
     pPayment->c_w_id++;
    }
   pPayment->c_d_id = random_int( 1, DISTRICTS_PER_WAREHOUSE
);
  }

  pPayment->h_amount = ((double)random_int( 100,
MAX_PT_AMOUNT_CENTS ) / 100.0);

}

void
get_os_data( txn_context_t *pContext, pOrderStatusData pOrderStatus,
                    int *context )
{
  char                                  *ptr;
  int                                   index, i1, i2, i3;

  pOrderStatus->CC = (CallersContext)pContext;

  *context = ORDER_STATUS_TXN;

  get_wdid( pContext, &pOrderStatus->w_id, &pOrderStatus->ld_id );

  pOrderStatus->d_id = random_int( 1, DISTRICTS_PER_WAREHOUSE );

  pOrderStatus->c_id = INVALID_C_ID;

  ptr = pOrderStatus->c_last;
  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
  {

    pOrderStatus->c_id = NURand( 1023, 1, MAX_C_ID, C_ID_CONST );
  }

}

void
get_dy_data( txn_context_t *pContext, pDeliveryData pDelivery,
                    int *context )
{
  pDelivery->CC = (CallersContext)pContext;

  *context = INT_DELIVERY_TXN;

  get_wdid( pContext, &pDelivery->w_id, &pDelivery->ld_id );


  pDelivery->o_carrier_id = random_int( 1, MAX_CARRIER_ID );


  pDelivery->queue_time = time( NULL );

}

void
get_sl_data( txn_context_t *pContext, pStockLevelData pStockLevel,
                    int *context )
{
  pStockLevel->CC = (CallersContext)pContext;

  *context = STOCK_LEVEL_TXN;

  get_wdid( pContext, &pStockLevel->w_id, &pStockLevel->ld_id );


  pStockLevel->threshold = random_int( MIN_SL_THRESHOLD,
MAX_SL_THRESHOLD );

}

void
get_cp_data( txn_context_t *pContext, pCheckpointData pCheckpoint,
                    int *context )
{
  pCheckpoint->CC = (CallersContext)pContext;

  *context = CHECKPOINT_TXN;

  get_wdid( pContext, &pCheckpoint->w_id, &pCheckpoint->ld_id );

  pCheckpoint->how_many = 1;
  pCheckpoint->interval = 0;

}
```

tpcc_gen.h

```
#ifndef TPCC_GEN_H
#define TPCC_GEN_H
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1999 BY                                   *
 *  COMPAQ COMPUTER CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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.                 *
 *                                                          *
 *                                                          *

************************************************************
************-*/

void get_wdid( txn_context_t *pContext, int *pw_id, int *pd_id );
void get_no_data(txn_context_t *pUdata, pNewOrderData pNewOrder, int
*context);
void get_pt_data(txn_context_t *pUdata, pPaymentData pPayment, int
*context);
void get_os_data(txn_context_t *pUdata, pOrderStatusData pOrderStatus,
                    int *context);
void get_dy_data(txn_context_t *pUdata, pDeliveryData pDelivery, int
*context);
void get_sl_data(txn_context_t *pUdata, pStockLevelData pStockLevel,
                    int *context);
void get_cp_data(txn_context_t *pUdata, pCheckpointData pCheckpoint,
                    int *context);

#endif
```

tpcc_master.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1999 BY                                   *
 *  COMPAQ COMPUTER CORPORATION, MAYNARD,
MASSACHUSETTS.                  *
 *  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 <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 <stdtypes.h>
#include <prte.h>
#include <common.h>

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

#include <scr_util.h>

#define NEED_MSG_NAMES
#include <msg.h>
#include <netvar.h>
#include <audit_util.h>


#ifdef _WIN32
# define MKDIR(Dir,Mode) mkdir(Dir)
  typedef int mode_t;
#else
# define MKDIR(Dir,Mode) mkdir(Dir,Mode)
#endif

#define MAX_RUN_DIR_NAME_LEN 3
#define LOG_DIR "logs"
#define maximum(A,B) ((A) > (B) ? (A) : (B))

#define NODELAY

#define REG_KEY_STR "CONF_FE_NT_REG_KEY_"
#define REG_VALUE_STR "CONF_FE_NT_REG_VALUE_"
#define REG_TYPE_STR "CONF_FE_NT_REG_TYPE_"
#define REG_DATA_STR "CONF_FE_NT_REG_DATA_"

typedef struct _be_t
{
  char          Name[MAXHOSTNAMELEN];
  double LogSize;
  double LogUsed;
} be_t;


typedef struct _master_data_t
{
  int32                 AdminUsers;
  int32                 RequestsSent;
  int32                 ReplysReceived;
  int32                 AidesStatus;
  BOOL                  MsgTimedOut;
  double        MsgTimeout;

  char                  CgiScriptName[PATH_MAX];
  BOOL                  DoCheckpoints;
  double        CkptStartOffset;


  int32                         RunMode;
  int32                         TpccUsers;
  int32                         NumFEs;
  int32                         NumConnects;
  connect_t             *pConnects;
  BOOL                          FERestartViaReboot;
  int32                         RestartFEs;

  double        Proximity;
  pid_t                 OsPid;
  char                  RunDir[PATH_MAX];
  int                   CheckpointInterval;
  double        WarmUpTime;
  double        SteadyStateTime;
  double        MeasurementInterval;
  double        CoolDownTime;
  int                   ReducerID;
  int32                 ReducerStatus;
  char                  ReducerScriptName[PATH_MAX];
  char                  ReducerScriptNode[MAXHOSTNAMELEN];
  char                  OutputDir[PATH_MAX];
  char                  RunNumStr[8];
  int                   VersionNum;
  long                  MasterSeed;
  char                  *pVersion;
  int32                 NumBE;
  be_t                  *pBEs;
  char                  AideScriptName[PATH_MAX];

  char                  BeUname[32];
  char                  DbUname[32];
  char                  DbPassword[32];
  char                  DbName[32];
  char                  OraStatScriptPath[PATH_MAX];
  char                  CustomBeforeTestScript[PATH_MAX];
  char                  CustomAfterTestScript[PATH_MAX];


  task_data_t   BeTasksData;
} master_data_t;
```

```
void            BinMsgHandler( void *Context, int SendersID, int Len, void
*Msg );
int32           Config( master_data_t *pContext );
int32           ConfigFEs( master_data_t *pContext );
int32           ConfigFEsSendMsg( master_data_t *pContext,
                                  config_fe_msg_t *pConfigFEMsg,
char *pDataVar );
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           InitAides( master_data_t *pContext );
int32           InitFEs( master_data_t *pContext );
int32           InitReducer( master_data_t *pContext );
int32           MakeRunDir(char *DirName, char *RunDir);
int32           ParseUsersPerFE(master_data_t *pContext, char *string);
int32   PreTestHook(master_data_t *pContext);
int32           StartFEs( master_data_t *pContext );
int32           StopAides( master_data_t *pContext );
int32           StopFEs( master_data_t *pContext );
int32           StopReducer( master_data_t *pContext );
void            TellAidesTo(master_data_t *pContext, int32 Task, int32 State);
int32           TestPeriod( master_data_t *pContext );


void
main(int argc, char *argv[])
{
  master_data_t                         Context;
  master_data_t                         *pContext = &Context;
  int32                                 Status;


  PRTEinit( argc, argv );


  if( SCR_E_SUCCESS != ( Status = Init( pContext )))
  {
    ( void ) SCRerr( Status );
    ( void ) SCRerr( SCR_E_INIT );
  }


  if( SCR_E_SUCCESS == Status )
  {
    switch( pContext->RunMode )
    {
    case END_TO_END_MODE:
    case BACKEND_MODE:
      Status = TestPeriod( pContext );
      break;
    case CONFIG_MODE:
      Status = Config( pContext );
      break;
    default:
      break;
    }
  }

  SCRlog( "Master: Blowing this popsicle stand." );

  PRTEexit( );
}


void
BinMsgHandler( void *Context, int SendersID, int Len, void *Msg )
{
  int32                                 Index;
  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( );
  break;

 case DATA_POST_PROCESS_REPLY_MSG:
  pStatusMsg = (generic_status_msg_t *) Msg;
  if( SCR_E_SUCCESS == pStatusMsg->Status )
  {
        SCRlog( "\tTest results successfully generated." );
  }
  else
  {
        SCRlog( "\tError generating test results ( %d ).",
                pStatusMsg->Status );
  }
  PRTEresume( );

  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( SCR_E_SUCCESS != pStatusMsg->Status &&
        SCR_E_SUCCESS == pContext->AidesStatus )
  {

        pContext->AidesStatus = pStatusMsg->Status;
  }
  if( pContext->ReplysReceived == pContext->RequestsSent )
  {

        pContext->MsgTimedOut = FALSE;
        PRTEdelay( 0.0 );
  }
  break;

 case AIDE_INIT_REPLY_MSG:
  pStatusMsg = (generic_status_msg_t *) Msg;
  if( SCR_E_SUCCESS != pStatusMsg->Status &&
        SCR_E_SUCCESS == pContext->AidesStatus )
  {

        pContext->AidesStatus = pStatusMsg->Status;
  }
  pContext->ReplysReceived++;
  if( pContext->ReplysReceived == pContext->RequestsSent )
  {

        pContext->MsgTimedOut = FALSE;
        PRTEdelay( 0.0 );
  }
  break;

 case CHECKPOINT_CONNECT_REPLY_MSG:
  pCheckpointStatusMsg = (checkpoint_status_msg_t *) Msg;
  pContext->ReplysReceived++;

  if( SCR_E_SUCCESS == pCheckpointStatusMsg->Data.Status )
  {
        SCRlog( "\t\t%s - checkpoint sub-system initialized.",
                pCheckpointStatusMsg->Data.FEName );
```

```
    }
    else
    {
            if( SCR_E_SUCCESS == pContext->AidesStatus )
            {
             pContext->AidesStatus = pCheckpointStatusMsg-
>Data.Status;
            }
            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( SCR_E_SUCCESS == pCheckpointStatusMsg->Data.Status )
    {
            SCRlog( "\t\t%s - checkpoint loop started.",
                        pCheckpointStatusMsg->Data.FEName );
    }
    else
    {
            if( SCR_E_SUCCESS == pContext->AidesStatus )
            {
             pContext->AidesStatus = pCheckpointStatusMsg-
>Data.Status;
            }
            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( SCR_E_SUCCESS == pCheckpointStatusMsg->Data.Status )
    {
            SCRlog( "\t\t%s - checkpoint loop ran successfully.",
                        pCheckpointStatusMsg->Data.FEName );
    }
    else
    {
            if( SCR_E_SUCCESS == pContext->AidesStatus )
            {
             pContext->AidesStatus = pCheckpointStatusMsg-
>Data.Status;
            }
            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 CHECKPOINT_DISCONNECT_REPLY_MSG:
    pCheckpointStatusMsg = (checkpoint_status_msg_t *) Msg;
    pContext->ReplysReceived++;

    if( SCR_E_SUCCESS == pCheckpointStatusMsg->Data.Status )
    {
            SCRlog( "\t\t%s - checkpoint sub-system shutdown.",
                        pCheckpointStatusMsg->Data.FEName );
    }
    else
    {
            if( SCR_E_SUCCESS == pContext->AidesStatus )
            {
             pContext->AidesStatus = pCheckpointStatusMsg-
>Data.Status;
            }
            SCRlog( "\t\t%s - checkpoint subsystem shutdown FAILED (
%d ).",
                        pCheckpointStatusMsg->Data.FEName,
                        pCheckpointStatusMsg->Data.Status );
    }

    if( pContext->ReplysReceived == pContext->RequestsSent )
    {

            pContext->MsgTimedOut = FALSE;
            PRTEdelay( 0.0 );
    }

    break;

   case CONFIG_FE_REPLY_MSG:
    pStatusMsg = (generic_status_msg_t *) Msg;
    pContext->ReplysReceived++;
    if( SCR_E_VALUE_MODIFIED == pStatusMsg->Status )
    {
            for( Index = 0; Index < pContext->NumConnects; Index++ )
            {
             if( SendersID == pContext->pConnects[Index].AideID )
             {
               pContext->pConnects[Index].ConfMod = TRUE;
               break;
             }
            }
    }
    else if( SCR_E_SUCCESS != pStatusMsg->Status &&
                SCR_E_SUCCESS == pContext->AidesStatus )
    {

            pContext->AidesStatus = pStatusMsg->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( SCR_E_SUCCESS != pStatusMsg->Status &&
              SCR_E_SUCCESS == pContext->AidesStatus )
    {

            pContext->AidesStatus = pStatusMsg->Status;
    }
    pContext->ReplysReceived++;
    if( pContext->ReplysReceived == pContext->RequestsSent )
    {

            pContext->MsgTimedOut = FALSE;
```

```
            PRTEdelay( 0.0 );
        }
        break;

    case START_FE_REPLY_MSG:
        pStatusMsg = (generic_status_msg_t *) Msg;
        pContext->ReplysReceived++;
        if( SCR_E_SUCCESS != pStatusMsg->Status &&
            SCR_E_SUCCESS == pContext->AidesStatus )
        {

            pContext->AidesStatus = pStatusMsg->Status;
        }
        if( pContext->ReplysReceived == pContext->RequestsSent )
        {

            pContext->MsgTimedOut = FALSE;
            PRTEdelay( 0.0 );
        }
        break;

    case STOP_FE_REPLY_MSG:
        pStatusMsg = (generic_status_msg_t *) Msg;
        pContext->ReplysReceived++;
        if( SCR_E_SUCCESS != pStatusMsg->Status &&
            SCR_E_SUCCESS == pContext->AidesStatus )
        {

            pContext->AidesStatus = pStatusMsg->Status;
        }
        if( pContext->ReplysReceived == pContext->RequestsSent )
        {

            pContext->MsgTimedOut = FALSE;
            PRTEdelay( 0.0 );
        }
        break;

    case REDUCER_INIT_REPLY_MSG:
        pStatusMsg = (generic_status_msg_t *) Msg;
        pContext->ReducerStatus = pStatusMsg->Status;
        pContext->MsgTimedOut = FALSE;
        PRTEdelay( 0.0 );
        break;

    case USER_LOGIN_REPLY_MSG:
    case USER_START_REPLY_MSG:
    case USER_STOP_REPLY_MSG:
        pStatusMsg = (generic_status_msg_t *) Msg;
        pContext->ReducerStatus = pStatusMsg->Status;
        PRTEresume( );
        break;

    case LOAD_FE_REPLY_MSG:
        pStatusMsg = (generic_status_msg_t *) Msg;
        pContext->ReplysReceived++;
        if( SCR_E_SUCCESS != pStatusMsg->Status &&
            SCR_E_SUCCESS == pContext->AidesStatus )
        {

            pContext->AidesStatus = pStatusMsg->Status;
        }
        if( pContext->ReplysReceived == pContext->RequestsSent )
        {

            pContext->MsgTimedOut = FALSE;
            PRTEdelay( 0.0 );
        }
        break;

    default:
        sprintf( TmpStr, "Master received unknown user message type: %d",
                 pMsg->Type );
        SCRlog( TmpStr );
    }
}



int32
Config( master_data_t *pContext )
{
    char                    *pTmpStr;
    int32                   Status;
    char                    TmpStr[256];

    Status = SCR_E_SUCCESS;

    SCRlog( "\n\n\n\t\t\t\t..START OF CONFIG..\n\n" );


    ITOA( 0, TmpStr, pTmpStr );
    PRTEset_network_variable( "REDUCER_ID", TmpStr );

    if( SCR_E_SUCCESS == Status )
    {
        Status = InitAides( pContext );
    }

    if( SCR_E_SUCCESS == Status )
    {
        Status = ConfigFEs( pContext );
    }

    if( SCR_E_SUCCESS == Status )
    {
        Status = StopAides( pContext );
    }

    SCRlog( "\n\n\n\n\t\t\t\t..END OF CONFIG..\n\n\n" );

    return( Status );
}


int32
ConfigFEs( master_data_t *pContext )
{
    int                     Code;
    int32                   Index;
    char                    NetVarName[50];
    int32                   NetVarNum;
    char                    *pTmp;
    char                    *pValue;
    config_fe_msg_t         ConfigFEMsg;
    int32                   Status;


    if( 0 == pContext->NumFEs )
    {
        return( SCR_E_SUCCESS );
    }


    for( Index = 0; Index < pContext->NumConnects; Index++ )
    {
        if( ! pContext->pConnects[Index].ConfConn )
        {
            continue;
        }
        pContext->pConnects[Index].ConfMod = FALSE;
        SCRlog( "\tChecking configuration of %s.",
                pContext->pConnects[Index].FEName );
    }


    strcpy( ConfigFEMsg.Data.Key, TPCC_CLASS );
    strcpy( ConfigFEMsg.Data.Type, "string" );

    for( Code = FFE_C_BASE + 1; Code <= FFE_C_MAX_PARAM; Code++
)
    {

        pValue = ConfigCodeToStr( Code );
        strcpy( ConfigFEMsg.Data.Value, pValue );


        Status = ConfigFEsSendMsg( pContext, &ConfigFEMsg, pValue );
        if( SCR_E_SUCCESS != Status )
        {
```

```
    return( Status );
   }
  }


  NetVarNum = 1;
  sprintf( NetVarName, "%s%d", REG_KEY_STR, NetVarNum );
  while( NULL != (pTmp = PRTEget_network_variable( NetVarName )))
  {


   strcpy( ConfigFEMsg.Data.Key, pTmp );


   sprintf( NetVarName, "%s%d", REG_VALUE_STR, NetVarNum );
   if( NULL == (pTmp = PRTEget_network_variable( NetVarName )))
   {

    SCRlog( "Getting network variable: %s", ConfigFEMsg.Data.Value );
    return( SCRerr( SCR_E_NO_VALUE ));
   }
   else
   {
    strcpy( ConfigFEMsg.Data.Value, pTmp );
   }


   sprintf( NetVarName, "%s%d", REG_TYPE_STR, NetVarNum );
   if( NULL == (pTmp = PRTEget_network_variable( NetVarName )))
   {

    SCRlog( "Getting network variable: %s", ConfigFEMsg.Data.Type );
    return( SCRerr( SCR_E_NO_TYPE ));
   }
   else
   {
    strcpy( ConfigFEMsg.Data.Type, pTmp );
   }


   sprintf( NetVarName, "%s%d", REG_DATA_STR, NetVarNum );
   Status = ConfigFEsSendMsg( pContext, &ConfigFEMsg, NetVarName );
   if( SCR_E_SUCCESS != Status )
   {
    return( Status );
   }


   NetVarNum++;
   sprintf( NetVarName, "%s%d", REG_KEY_STR, NetVarNum );
  }


  Status = SCR_E_SUCCESS;
  for( Index = 0; Index < pContext->NumConnects; Index++ )
  {
   if( ! pContext->pConnects[Index].ConfConn )
   {
    continue;
   }
   if( pContext->pConnects[Index].ConfMod )
   {
    Status = SCR_E_VALUE_MODIFIED;
    SCRlog( "\tSuccessfully modified configuration of %s.",
                pContext->pConnects[Index].FEName );
   }
   else
   {
    SCRlog( "\tSuccessfully checked configuration of %s.",
                pContext->pConnects[Index].FEName );
   }
  }

  if( SCR_E_VALUE_MODIFIED != Status && SCR_E_SUCCESS !=
Status )
  {
   SCRlog( "\tFailed to configure the FEs. Pausing.\n" );
   PRTEpause();
  }

  if( FE_RESTART_ALWAYS == pContext->RestartFEs ) ||
    ( FE_RESTART_MODIFIED == pContext->RestartFEs &&
      SCR_E_VALUE_MODIFIED == Status ))
  {
   if( SCR_E_SUCCESS != ( Status = StopFEs( pContext )))
   {
    SCRlog( "\tFailed to stop the FEs. Pausing.\n" );
    PRTEpause();
   }
   if( SCR_E_SUCCESS != ( Status = StartFEs( pContext )))
   {
    SCRlog( "\tFailed to start the FEs. Pausing.\n" );
    PRTEpause();
   }
  }
  else
  {

   Status = SCR_E_SUCCESS;
  }

  return( Status );
}


int32
ConfigFEsSendMsg( master_data_t *pContext, config_fe_msg_t
*pConfigFEMsg,
                  char *pDataVar )
{
  int                   Count;
  int32                 Index;
  char                  *pTmp;
  int32                 Status;

  Status = SCR_E_SUCCESS;
  if( NULL == (pTmp = PRTEget_network_variable( pDataVar )))
  {

   SCRlog( "Getting network variable: %s", pDataVar );
   return( SCRerr( SCR_E_NO_DATA ));
  }


  Count = SCR_count_elements( pTmp );
  if( 1 != Count && pContext->NumFEs != Count )
  {
   SCRlog( "Parsing: %s %s", pDataVar, pTmp );
   return( SCRerr( SCR_E_WRONG_DATA_ITEM_COUNT ));
  }


  pContext->RequestsSent = 0;
  pContext->ReplysReceived = 0;
  pContext->AidesStatus = SCR_E_SUCCESS;
  pContext->MsgTimedOut = TRUE;
  pConfigFEMsg->Type = CONFIG_FE_REQUEST_MSG;
  for( Index = 0; Index < pContext->NumConnects; Index++ )
  {
   if( ! pContext->pConnects[Index].ConfConn )
   {
    continue;
   }
   if( 1 == Count )
   {


    (void)SCR_next_list_element( pTmp, pConfigFEMsg->Data.Data );
   }
   else
   {

    pTmp = SCR_next_list_element( pTmp, pConfigFEMsg->Data.Data );
   }
   strcpy( pConfigFEMsg->Data.FEName, pContext-
>pConnects[Index].FEName );
   PRTEmessage_to_user_binary( sizeof( *pConfigFEMsg ),
pConfigFEMsg,
```

```
                                                pContext-
>pConnects[Index].AideID,
                                                pContext-
>pConnects[Index].AideID );
    pContext->RequestsSent++;
   }

   PRTEdelay( pContext->MsgTimeout );

   if( pContext->MsgTimedOut )
   {
    Status = SCRerr( SCR_E_AIDE_TIMEOUT_CONFIG_FE );
    SCRlog( "\tPausing...\n" );
    PRTEpause( );
   }

   if( SCR_E_SUCCESS != pContext->AidesStatus )
   {
    Status = SCRerr( pContext->AidesStatus );
    Status = SCRerr( SCR_E_AIDES_FAILED_CONFIG );
   }

   return( Status );
}


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( SCR_E_BE_NAMES ));
   }

   pContext->NumBE = SCR_count_elements(List);


   if( NULL == ( pContext->pBEs =
                        ( be_t * ) malloc( pContext->NumBE * sizeof(
be_t ))))
   {
    return( SCRerr( SCR_E_MALLOC_BE_ARRAY ));
   }

   SCRlog( "\tThe following BEs will be used." );
   pTmpChar = List;
   BeIndex = 0;
   while( pTmpChar = SCR_next_list_element( pTmpChar, ElementString ))
   {
    strcpy( pContext->pBEs[BeIndex].Name, ElementString );
    SCRlog( "\t\t%s", pContext->pBEs[BeIndex].Name );
    BeIndex++;
   }

   return( SCR_E_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;
   FILE                    *Fd;
   char                    FileName[PATH_MAX];
   char                    IDString[7];
   int                     Index;
   int32                   RunId;
   int32                   Status;
   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( SCR_E_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( SCR_E_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 );
  }


  Status = GetConnectInfo( pContext->RunMode, &pContext->TpccUsers,
                           &pContext->NumFEs, &pContext-
>AdminUsers,
                           &pContext->NumConnects,
&pContext->pConnects );
  if( SCR_E_SUCCESS != Status )
  {
   SCRlog( "Error ( %d ) getting CONNECT information.", Status );
   return( Status );
  }


  SCR_getnet_int( "RUN_MODE", RUN_MODE_DEFAULT,
                  &pContext->RunMode );
  switch( pContext->RunMode )
  {
   case END_TO_END_MODE:
    SCRlog( "\tRunning in end-to-end mode.\n" );
    break;
   case BACKEND_MODE:
    SCRlog( "\tRunning in rte based backend run (C') mode.\n" );
    break;
   case CONFIG_MODE:
    SCRlog( "\tRunning in config mode.\n" );
    break;
   default:
    SCRlog( "RunMode = %d", pContext->RunMode );
    return( SCRerr( SCR_E_INVALID_RUN_MODE ));
    break;
  }

  SCRlog( "\tConnects." );
#define CONNECT_TABLE_HEADER_FORMAT "\t  %-12s %-12s %-5s
%-5s %5s %s"
#define CONNECT_TABLE_ENTRY_FORMAT  "\t  %-12s %-12s %-5s
%-5s %5d %s"
  SCRlog( CONNECT_TABLE_HEADER_FORMAT,
          "RTE", "FE", "Conf.", "Admin", "Users", "Script");
  for( Index = 0; Index < pContext->NumConnects; Index++ )
  {
   SCRlog( CONNECT_TABLE_ENTRY_FORMAT,
           (pContext->pConnects)[Index].RTEName,
           (pContext->pConnects)[Index].FEName,
           (pContext->pConnects)[Index].ConfConn ? "  T" : "  F",
           (pContext->pConnects)[Index].AdminConn ? "  T" : "  F",
           (pContext->pConnects)[Index].Ucnt,
           (pContext->pConnects)[Index].Script );
  }
  SCRlog( "\tTotal TPC-C Users = %d.\n", pContext->TpccUsers );
  SCRlog( "\tTotal FE Count = %d.\n", pContext->NumFEs );
  SCRlog( "\tTotal Admin Users = %d.\n", pContext->AdminUsers );


  SCR_getnet_bool( "FE_RESTART_VIA_REBOOT",
FE_RESTART_VIA_REBOOT_DEFAULT,
                   &pContext->FERestartViaReboot );
  SCR_getnet_int( "FE_RESTART_MODE",
FE_RESTART_MODE_DEFAULT,
                  &pContext->RestartFEs );


  SCR_getnet_string( "ADMIN_SCRIPT_NAME",
ADMIN_SCRIPT_NAME_DEFAULT,
                     pContext->AideScriptName );
  SCRlog( "\tAide Script Name = %s.", pContext->AideScriptName );

  SCR_getnet_string( "REDUCER_SCRIPT_NAME",
REDUCER_SCRIPT_NAME_DEFAULT,
                     pContext->ReducerScriptName );
  SCRlog( "\tReducer Script Name = %s.", pContext->ReducerScriptName );
  SCR_getnet_string( "REDUCER_SCRIPT_NODE",
REDUCER_SCRIPT_NODE_DEFAULT,
                     pContext->ReducerScriptNode );
```

```
 SCRlog( "\tReducer Script Node = %s.", pContext->ReducerScriptNode );



 SCR_getnet_string( "CGI_SCRIPT_NAME",
CGI_SCRIPT_NAME_DEFAULT,
                        pContext->CgiScriptName );

 if( 0 < pContext->AdminUsers )
 {

  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( SCR_E_CKPT_INT_TOO_LONG ));
   }


   if( pContext->CheckpointInterval > pContext->MeasurementInterval )
   {
        return( SCRerr(
SCR_E_CKPT_INT_GREATER_THAN_MEASUREMENT_INT ));
   }



   if( (pContext->CheckpointInterval < pContext->MeasurementInterval)
&&
        ( (int32) pContext->MeasurementInterval %
          pContext->CheckpointInterval != 0 ))
   {
        return( SCRerr(
SCR_E_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( SCR_E_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." );
  }
 }


#pragma message("DJ - FixMe: Should this check be, or include,
GET_ALL_AUDIT_FILES? Right now, it will print out that Audit utilities
will be run, even if GET_ALL_AUDIT_FILES is set to false. This seems
contradictory." )
 if( 0 < pContext->AdminUsers )
 {
  SCRlog( "\tAudit utilities will be run." );


  if( SCR_E_SUCCESS != ( Status = GetBeInfo( pContext )))
  {
   ( void ) SCRerr( SCR_E_GET_BE_INFO );
   return( Status );
  }

  Status = UTILGetAuditFunctionNetworkVariables( &pContext-
>BeTasksData );
  if( SCR_E_SUCCESS != Status )
  {
   SCRlog( "Error getting audit function net vars." );
   return( Status );
  }
  else
  {
   SCRlog( "\t\tDatabase type is %s", pContext->BeTasksData.DbTypeStr
);
  }
 }

 return( SCR_E_SUCCESS );
}


int32
InitAides( master_data_t *pContext )
{
 int32                  AideID;
 int32                  Index;
 int32                  Status;

 Status = SCR_E_SUCCESS;

 if( 0 < pContext->NumFEs || 0 < pContext->AdminUsers )
 {
  SCRlog( "\tAides being created and initialized.\n\n" );

  pContext->RequestsSent = 0;
  pContext->ReplysReceived = 0;
  pContext->AidesStatus = SCR_E_SUCCESS;
  pContext->MsgTimedOut = FALSE;

  for( Index = 0; Index < pContext->NumConnects; Index++ )
  {
   if( ! pContext->pConnects[Index].ConfConn &&
        ! pContext->pConnects[Index].AdminConn )
   {
        pContext->pConnects[Index].AideID = 0;
        continue;
   }
   AideID = PRTExecute_script( pContext->AideScriptName,
                        pContext-
>pConnects[Index].RTEName,
```

```
                                                    1, pContext-
>pConnects[Index].FEName, 0.0 );
      if( -1 == AideID )
      {
                Status = SCRerr( SCR_E_AIDE_CREATE_FAILED );
                SCRlog( "\tPausing.\n" );
                PRTEpause();
                return( Status );
      }

      PRTEstart_users( AideID, AideID );


      pContext->pConnects[Index].AideID = AideID;
      pContext->RequestsSent++;
    }

    pContext->MsgTimedOut = TRUE;
    PRTEdelay( pContext->MsgTimeout );
    SCRlog( "Master: Resuming.\n" );

    if( pContext->MsgTimedOut )
    {

      Status = SCRerr( SCR_E_AIDE_INIT_TIMEOUT );
      SCRlog( "\tPausing indefinitely....\n\n" );
      PRTEpause();
    }

    if( SCR_E_SUCCESS != pContext->AidesStatus )
    {
      Status = SCRerr( pContext->AidesStatus );
      SCRlog( "\tNot all aides initialized correctly. Pausing.\n" );
      PRTEpause();
    }
    else
    {
      SCRlog( "\tAides successfully initialized.\n" );
    }
  }

  return( Status );
}


int32
InitFEs( master_data_t *pContext )
{
  int32                                         Index;
  load_fe_msg_t                                 LoadFEMsg;
  int32                                         Status;


  if( 0 == pContext->NumFEs )
  {
    return( SCR_E_SUCCESS );
  }


  SCR_getnet_int( "LOAD_FE_TIMEOUT",
LOAD_FE_TIMEOUT_DEFAULT,
                        &LoadFEMsg.Data.Timeout );
  SCRlog( "\tFE(s) being loaded.  FE Load Timeout = %d\n",
                LoadFEMsg.Data.Timeout );

  LoadFEMsg.Type = LOAD_FE_REQUEST_MSG;

  pContext->RequestsSent = 0;
  pContext->ReplysReceived = 0;
  pContext->AidesStatus = SCR_E_SUCCESS;
  pContext->MsgTimedOut = TRUE;

  for( Index = 0; Index < pContext->NumConnects; Index++)
  {
    if( ! pContext->pConnects[Index].ConfConn )
    {
      continue;
    }

    strcpy( LoadFEMsg.Data.FEName, pContext-
>pConnects[Index].FEName );
#pragma message ("FIXME: the cgi script (dll) name is possibly different.
This code should parse a comma separated list?  Not here, but where the
CgiScriptName is originally loaded.")
    strcpy( LoadFEMsg.Data.CgiScriptName, pContext->CgiScriptName );

    PRTEmessage_to_user_binary( sizeof( LoadFEMsg ), &LoadFEMsg,
                                                    pContext-
>pConnects[Index].AideID,
                                                    pContext-
>pConnects[Index].AideID );
    pContext->RequestsSent++;
  }

  PRTEdelay( LoadFEMsg.Data.Timeout );

  Status = SCR_E_SUCCESS;
  if( pContext->MsgTimedOut )
  {

    Status = SCRerr( SCR_E_LOAD_FE_TIMEOUT );
  }
  else if( SCR_E_SUCCESS != pContext->AidesStatus )
  {
    Status = SCRerr( pContext->AidesStatus );
    Status = SCRerr( SCR_E_LOAD_FE_FAILURE );
  }
  else
  {
    SCRlog( "\tFE(s) successfully loaded.\n" );
  }

  if( SCR_E_SUCCESS != Status )
  {
    SCRlog( "\tFailed to load FEs' application code. Pausing.\n" );
    PRTEpause();
  }

  return( Status );
}


int32
InitReducer( master_data_t *pContext )
{
  int32                                 Status;

  Status = SCR_E_SUCCESS;



  SCRlog( "\tReducer being created.\n\n" );

  pContext->ReducerStatus = SCR_E_SUCCESS;
  pContext->MsgTimedOut = FALSE;

  pContext->ReducerID = PRTEexecute_script( pContext-
>ReducerScriptName,
                                                                    pContext-
>ReducerScriptNode,
                                                                    1, "", 0.0
);
  if( -1 == pContext->ReducerID )
  {
    Status = SCRerr( SCR_E_REDUCER_CREATE_FAILED );
    SCRlog( "\tPausing.\n" );
    PRTEpause();
  }
  else
  {
    SCRlog( "\tReducer being told to initialize.\n\n" );
    PRTEstart_users( pContext->ReducerID, pContext->ReducerID );

    pContext->MsgTimedOut = TRUE;
    PRTEdelay( pContext->MsgTimeout );
    SCRlog( "Master: Resuming.\n" );
  }

  if( pContext->MsgTimedOut )
```

```
  {

    Status = SCRerr( SCR_E_REDUCER_INIT_TIMEOUT );
    SCRlog( "\tPausing.\n" );
    PRTEpause();
  }

  if( SCR_E_SUCCESS != pContext->ReducerStatus )
  {
    SCRlog( "\tReducer failed to initialize correctly. Pausing.\n" );
    PRTEpause();
  }
  else
  {
    SCRlog( "\tReducer successfully initialized.\n" );
  }

  return( Status );
}

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;
}


int32
StartFEs( master_data_t *pContext )
{
  int32                 Index;
  start_fe_msg_t        StartFEMsg;
  start_fe_msg_t        *pStartFEMsg = &StartFEMsg;
  double                StartTimeout;
  int32                 Status;

  Status = SCR_E_SUCCESS;
  if( 0 == pContext->NumFEs )
  {
    return( SCR_E_SUCCESS );
  }

#pragma message ("FIXME: Should there be a separate start timeout as
network variable?")
  StartTimeout = pContext->MsgTimeout;

  pContext->RequestsSent = 0;
  pContext->ReplysReceived = 0;
  pContext->AidesStatus = SCR_E_SUCCESS;
  pContext->MsgTimedOut = TRUE;
  pStartFEMsg->Type = START_FE_REQUEST_MSG;
  pStartFEMsg->Data.Timeout = StartTimeout;
  for( Index = 0; Index < pContext->NumConnects; Index++ )
  {
    if( ! pContext->pConnects[Index].ConfConn )
    {
      continue;
```

```
    }
    else if( FE_RESTART_ALWAYS != pContext->RestartFEs &&
                    ! pContext->pConnects[Index].ConfMod )
    {
      continue;
    }
    pStartFEMsg->Data.ConnectCount = pContext->pConnects[Index].Ucnt;
    strcpy( pStartFEMsg->Data.FEName, pContext-
>pConnects[Index].FEName );
    PRTEmessage_to_user_binary( sizeof( *pStartFEMsg ), pStartFEMsg,
                                                        pContext-
>pConnects[Index].AideID,
                                                        pContext-
>pConnects[Index].AideID );
    pContext->RequestsSent++;
    SCRlog( "\tStarting %s.", pContext->pConnects[Index].FEName );
  }


  if( 0 == pContext->RequestsSent )
  {
    return( SCR_E_SUCCESS );
  }


  PRTEdelay( StartTimeout + pContext->MsgTimeout );

  if( pContext->MsgTimedOut )
  {
    Status = SCRerr( SCR_E_AIDE_TIMEOUT_START_FE );
    SCRlog( "Pausing...\n" );
    PRTEpause( );
  }

  if( SCR_E_SUCCESS != pContext->AidesStatus )
  {
    Status = SCRerr( pContext->AidesStatus );
    Status = SCRerr( SCR_E_AIDES_FAILED_START );
  }
  else
  {
    for( Index = 0; Index < pContext->NumConnects; Index++ )
    {
      if( ! pContext->pConnects[Index].ConfConn )
      {
                continue;
      }
      else if( FE_RESTART_ALWAYS != pContext->RestartFEs &&
                      ! pContext->pConnects[Index].ConfMod )
      {
                continue;
      }
      SCRlog( "\tSuccessfully started %s.",
                      pContext->pConnects[Index].FEName );
    }
  }

  return( Status );
}


int32
StopAides( master_data_t *pContext )
{
  generic_msg_t                 GenericMsg;
  int32                         Index;
  int32                         Status;

  Status = SCR_E_SUCCESS;

  if( 0 < pContext->NumFEs || 0 < pContext->AdminUsers )
  {
    SCRlog( "\tAides being told to exit." );
    pContext->RequestsSent = 0;
    GenericMsg.Type = AIDE_EXIT_REQUEST_MSG;
    pContext->ReplysReceived = 0;
    pContext->AidesStatus = SCR_E_SUCCESS;
    pContext->MsgTimedOut = TRUE;

    for( Index = 0; Index < pContext->NumConnects; Index++ )
    {
      if( ! pContext->pConnects[Index].ConfConn )
      {
                continue;
      }
      PRTEmessage_to_user_binary( sizeof( GenericMsg ), &GenericMsg,
                                                        pContext-
>pConnects[Index].AideID,
                                                        pContext-
>pConnects[Index].AideID );
      pContext->RequestsSent++;
    }

    PRTEdelay( pContext->MsgTimeout );

    if( pContext->MsgTimedOut )
    {
      Status = SCRerr( SCR_E_AIDE_EXIT_TIMEOUT );
    }

    if( SCR_E_SUCCESS != pContext->AidesStatus )
    {
      Status = SCRerr( pContext->AidesStatus );
      SCRlog( "Not all aides exited successfully." );
    }
    else
    {
      SCRlog( "\t%d Aides exited successfully.\n", pContext-
>ReplysReceived );
    }
  }

  return( Status );
}


int32
StopFEs( master_data_t *pContext )
{
  int32                         Index;
  stop_fe_msg_t                 StopFEMsg;
  stop_fe_msg_t                 *pStopFEMsg = &StopFEMsg;
  double                StopTimeout;
  int32                         Status;

  Status = SCR_E_SUCCESS;
  if( 0 == pContext->NumFEs )
  {
    return( SCR_E_SUCCESS );
  }

#pragma message ("FIXME: Should there be a separate stop timeout as
network variable?")
  StopTimeout = pContext->MsgTimeout;

  pContext->RequestsSent = 0;
  pContext->ReplysReceived = 0;
  pContext->AidesStatus = SCR_E_SUCCESS;
  pContext->MsgTimedOut = TRUE;
  pStopFEMsg->Type = STOP_FE_REQUEST_MSG;
  pStopFEMsg->Data.Timeout = StopTimeout;
  pStopFEMsg->Data.Reboot = pContext->FERestartViaReboot;
  for( Index = 0; Index < pContext->NumConnects; Index++ )
  {
    if( ! pContext->pConnects[Index].ConfConn )
    {
      continue;
    }
    else if( FE_RESTART_ALWAYS != pContext->RestartFEs &&
                    ! pContext->pConnects[Index].ConfMod )
    {
      continue;
    }
    strcpy( pStopFEMsg->Data.FEName, pContext-
>pConnects[Index].FEName );
    PRTEmessage_to_user_binary( sizeof( *pStopFEMsg ), pStopFEMsg,
                                                        pContext-
>pConnects[Index].AideID,
                                                        pContext-
>pConnects[Index].AideID );
```

```
    pContext->RequestsSent++;
    SCRlog( "\tStoping %s.", pContext->pConnects[Index].FEName );
   }


   if( 0 == pContext->RequestsSent )
   {
    return( SCR_E_SUCCESS );
   }


   PRTEdelay( StopTimeout + pContext->MsgTimeout );

   if( pContext->MsgTimedOut )
   {
    Status = SCRerr( SCR_E_AIDE_TIMEOUT_STOP_FE );
    SCRlog( "Pausing...\n" );
    PRTEpause( );
   }

   if( SCR_E_SUCCESS != pContext->AidesStatus )
   {
    Status = SCRerr( pContext->AidesStatus );
    Status = SCRerr( SCR_E_AIDES_FAILED_STOP );
   }
   else
   {
    for( Index = 0; Index < pContext->NumConnects; Index++ )
    {
     if( ! pContext->pConnects[Index].ConfConn )
     {
                continue;
     }
     else if( FE_RESTART_ALWAYS != pContext->RestartFEs &&
                        ! pContext->pConnects[Index].ConfMod )
     {
                continue;
     }
     SCRlog( "\tSuccessfully stopped %s.",
                        pContext->pConnects[Index].FEName );
    }
   }

   return( Status );
}


int32
StopReducer( master_data_t *pContext )
{

   return( SCR_E_SUCCESS );
}


void
TellAidesTo( master_data_t *pContext, int32 Task, int32 State )
{
   int32 BeIndex;
   int32 Index;
   generic_dotask_msg_t SendMsg;

   if (pContext->AdminUsers == 0)
    return;


   SCRlog( "\t%d aides being told to %s.", pContext->AdminUsers,
MsgNames[Task]);
   SendMsg.Type = Task;
   SendMsg.Data.State = State;

   pContext->RequestsSent = 0;
   pContext->ReplysReceived = 0;
   pContext->AidesStatus = SCR_E_SUCCESS;
   pContext->MsgTimedOut = TRUE;

   for( Index = 0, BeIndex = 0; Index < pContext->NumConnects; Index++ )
   {
    if( ! pContext->pConnects[Index].AdminConn )
    {
     continue;
    }
    SendMsg.Data.HostId = BeIndex + 1;
    strcpy( SendMsg.Data.SutName, pContext->pBEs[BeIndex].Name );
    PRTEmessage_to_user_binary( sizeof( SendMsg ), &SendMsg,
                                                pContext-
>pConnects[Index].AideID,
                                                pContext-
>pConnects[Index].AideID );
    pContext->RequestsSent++;
    BeIndex++;
   }

   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( SCR_E_SUCCESS != pContext->AidesStatus )
   {
    ( void )SCRerr( pContext->AidesStatus );
    SCRlog( "\tAides failed to %s.", MsgNames[Task] );
    SCRlog( "\tPausing.\n\n" );
    PRTEpause();
   }
   else
   {
    SCRlog( "\t%d aides successfully did %s.\n", pContext->AdminUsers,
                MsgNames[Task]);
   }

   return;
}


int32
TestPeriod( master_data_t *pContext )
{
   checkpoint_connect_msg_t        CheckpointConnect;
   checkpoint_run_msg_t            CheckpointRun;
   checkpoint_disconnect_msg_t     CheckpointDisconnect;
   generic_msg_t                           GenericMsg;
   int32                           Index;
   measurement_info_msg_t          Measurement;
   double                          TimeStamp;
   double                          TimeStamp2;
   double                          DelayTime;
   double                          StartTime;
   int32                           Status;


   SCRlog( "\n\n\n\t\t\t\t..START OF TEST..\n\n" );


   Status = InitReducer( pContext );


   Status = InitAides( pContext );


   Status = ConfigFEs( pContext );


   Status = InitFEs( pContext );


   if( pContext->DoCheckpoints )
   {
    TellAidesTo( pContext, BE_CKPTDATA_REQUEST_MSG, BEFORE );
   }


   if( pContext->BeTasksData.DbType == DB_TYPE_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->ReducerStatus = SCR_E_SUCCESS;
  PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
  PRTEpause();


  SCRlog("Master: Resuming.\n");

  if( SCR_E_SUCCESS != pContext->ReducerStatus )
  {
   SCRlog( "\tUsers failed to log in successfully. Pausing.\n" );
   PRTEpause();
  }
  else
  {
   SCRlog( "\tAll users logged in successfully.\n" );
  }


  if( pContext->DoCheckpoints )
  {
   SCRlog( "\t%d Aides initializing checkpoint sub-systems.",
                pContext->AdminUsers );
   CheckpointConnect.Type =
CHECKPOINT_CONNECT_REQUEST_MSG;
   strcpy( CheckpointConnect.Data.Dll, pContext->CgiScriptName );
   pContext->RequestsSent = 0;
   pContext->ReplysReceived = 0;
   pContext->AidesStatus = SCR_E_SUCCESS;
   pContext->MsgTimedOut = TRUE;

   for( Index = 0; Index < pContext->NumConnects; Index++ )
   {
    if( ! pContext->pConnects[Index].AdminConn )
    {
            continue;
    }
    strcpy( CheckpointConnect.Data.FEName, pContext-
>pConnects[Index].FEName);
     PRTEmessage_to_user_binary( sizeof( CheckpointConnect ),
                                                &CheckpointConnect,
                                                pContext-
>pConnects[Index].AideID,
                                                pContext-
>pConnects[Index].AideID );
     pContext->RequestsSent++;
   }

   PRTEdelay( pContext->MsgTimeout );

   if( pContext->MsgTimedOut )
   {
    ( void ) SCRerr( SCR_E_AIDE_CKPT_INIT_TIMEOUT );
    SCRlog( "Pausing indefinitely...\n\n" );
    PRTEpause();
   }

   if( SCR_E_SUCCESS != pContext->AidesStatus )
   {
    Status = SCRerr( pContext->AidesStatus );
    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->ReducerStatus = SCR_E_SUCCESS;
  PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
  PRTEpause();


  SCRlog("Master: Resuming.\n");

  if( SCR_E_SUCCESS != pContext->ReducerStatus )
  {
   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->AdminUsers );


   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 = 0;
   pContext->ReplysReceived = 0;
   pContext->AidesStatus = SCR_E_SUCCESS;
   pContext->MsgTimedOut = TRUE;

   for( Index = 0; Index < pContext->NumConnects; Index++ )
   {
    if( ! pContext->pConnects[Index].AdminConn )
    {
            continue;
    }
    strcpy( CheckpointRun.Data.FEName, pContext-
>pConnects[Index].FEName );
    PRTEmessage_to_user_binary( sizeof( CheckpointRun ),
&CheckpointRun,
                                                pContext-
>pConnects[Index].AideID,
                                                pContext-
>pConnects[Index].AideID );
    pContext->RequestsSent++;
   }

   PRTEdelay( pContext->MsgTimeout );

   if( pContext->MsgTimedOut )
   {
    ( void ) SCRerr( SCR_E_AIDE_CKPT_RUN_TIMEOUT );
```

```
    SCRlog( "Pausing indefinitely...\n\n" );
    PRTEpause();
   }
   else if( SCR_E_SUCCESS != pContext->AidesStatus )
   {
    Status = SCRerr( pContext->AidesStatus );
    SCRlog( "Not all aides entered checkpoint loop.." );
    SCRlog( "Pausing...\n\n" );
    PRTEpause();
   }
   else
   {
    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->AdminUsers );

   GenericMsg.Type = CHECKPOINT_STOP_REQUEST_MSG;

   pContext->RequestsSent = 0;
   pContext->ReplysReceived = 0;
   pContext->AidesStatus = SCR_E_SUCCESS;
   pContext->MsgTimedOut = TRUE;

   for( Index = 0; Index < pContext->NumConnects; Index++ )
   {
    if( ! pContext->pConnects[Index].AdminConn )
    {
            continue;
    }
    PRTEmessage_to_user_binary( sizeof( GenericMsg ), &GenericMsg,
                                                pContext-
>pConnects[Index].AideID,
                                                pContext-
>pConnects[Index].AideID );
    pContext->RequestsSent++;
   }

   PRTEdelay( pContext->MsgTimeout );

   if( pContext->MsgTimedOut )
   {
    ( void ) SCRerr( SCR_E_AIDE_CKPT_STOP_TIMEOUT );
    SCRlog( "Pausing indefinitely...\n\n" );
    PRTEpause();
   }

   if( SCR_E_SUCCESS != pContext->AidesStatus )
   {
    Status = SCRerr( pContext->AidesStatus );
    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();


  SCRlog("\tPausing while users stop doing transactions.\n\n");
  pContext->ReducerStatus = SCR_E_SUCCESS;
  PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
  PRTEpause();

  SCRlog("Master: Resuming.\n");

  if( SCR_E_SUCCESS != pContext->ReducerStatus )
  {
   SCRlog( "\tUsers failed to stop successfully. Pausing.\n" );
   PRTEpause();
  }
  else
  {
   SCRlog( "\tAll users successfully stopped doing transactions.\n" );
  }



  if( pContext->DoCheckpoints )
  {
   SCRlog( "\t%d Aides disconnecting checkpoint sub-systems.",
                pContext->AdminUsers );
   CheckpointDisconnect.Type =
CHECKPOINT_DISCONNECT_REQUEST_MSG;
   strcpy( CheckpointDisconnect.Data.Dll, pContext->CgiScriptName );
   pContext->RequestsSent = 0;
   pContext->ReplysReceived = 0;
   pContext->AidesStatus = SCR_E_SUCCESS;
   pContext->MsgTimedOut = TRUE;

   for( Index = 0; Index < pContext->NumConnects; Index++ )
   {
    if( ! pContext->pConnects[Index].AdminConn )
    {
            continue;
    }
    strcpy( CheckpointDisconnect.Data.FEName,
                pContext->pConnects[Index].FEName );
    PRTEmessage_to_user_binary( sizeof( CheckpointDisconnect ),

&CheckpointDisconnect,
                                                pContext-
>pConnects[Index].AideID,
                                                pContext-
>pConnects[Index].AideID );
    pContext->RequestsSent++;
   }
```

```
  PRTEdelay( pContext->MsgTimeout );

  if( pContext->MsgTimedOut )
  {
   ( void ) SCRerr( SCR_E_AIDE_CKPT_DISCONNECT_TIMEOUT );
   SCRlog( "Pausing indefinitely...\n\n" );
   PRTEpause();
  }

  if( SCR_E_SUCCESS != pContext->AidesStatus )
  {
   Status = SCRerr( pContext->AidesStatus );
   SCRlog( "Not all aides did checkpoint disconnect successfully." );
   SCRlog( "Pausing...\n\n" );
   PRTEpause();
  }

  SCRlog( "\t%d Aides successfully disconnected checkpoint sub-
systems.\n",
                pContext->ReplysReceived );
 }


 SCRlog("\tPost process binary logfile.\n\n");

 pContext->ReducerStatus = SCR_E_SUCCESS;
 PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
 PRTEpause();

 SCRlog("Master: Resuming.\n");


 if( pContext->DoCheckpoints )
 {
  TellAidesTo( pContext, BE_CKPTDATA_REQUEST_MSG, AFTER );
 }


 if( pContext->BeTasksData.GetAllAuditFiles )
 {
  TellAidesTo( pContext, GET_LOGSIZE_REQUEST_MSG, AFTER );
 }

 if( pContext->BeTasksData.DbType == DB_TYPE_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->ReducerStatus = SCR_E_SUCCESS;
 PRTEresume_users( pContext->ReducerID, pContext->ReducerID );
 PRTEpause();

 SCRlog("Master: Resuming.\n");



 Status = StopAides( pContext );

 Status = StopReducer( pContext );

 free( pContext->pConnects );

 SCRlog( "\n\n\n\n\t\t\t\t..END OF TEST..\n\n\n" );

#pragma message ("FIXME: This routine needs to keep a running track of
the status of all operations.")

 return( Status );
}
```

tpcc_user.c

```
/*+**********************************************************
**************
 *                                                          *
 *  COPYRIGHT (c) 1999 BY                                   *
 *  COMPAQ COMPUTER CORPORATION, MAYNARD,
MASSACHUSETTS.                *
 *  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 <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 <stdtypes.h>
#include <prte.h>
#include <common.h>

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

#include <scr_util.h>

#include <msg.h>
#include <netvar.h>
#include <tx_api.h>


#define MAXSTRINGSIZE   4096

typedef struct _user_context_t {
  long                                  myseed;
  char                                  *Version;
  txn_context_t                         TxnContext;
} user_context_t;


void BinMsgHandler(void *Context, int SendersID, int Len, void *Msg);
```

```
void DisconnectExit(user_context_t *pContext, int status, int Disconnect);
int  Init(user_context_t *pContext);



void
main(int argc, char *argv[])
{
  generic_status_msg_t    StatusMsg;
  user_login_reply_msg_t LoginMsg;
  user_context_t          Context;
  user_context_t          *pContext = &Context;


  PRTEinit(argc,argv);



  PRTEcatch_message_binary(BinMsgHandler, pContext);


  pContext->TxnContext.TxnTiming.Data.MenuStart = 0.0;
  pContext->TxnContext.TxnTiming.Data.MenuResponse = 0.0;
  pContext->TxnContext.TxnTiming.Data.TxnStart = 0.0;
  pContext->TxnContext.TxnTiming.Data.TxnResponse = 0.0;


  pContext->TxnContext.ReducerID =
    atoi( PRTEwait_for_network_variable( "REDUCER_ID" ));
  StatusMsg.Status = SCR_E_SUCCESS;
  StatusMsg.Type = USER_INIT_REPLY_MSG;
  PRTEmessage_to_user_binary( sizeof( StatusMsg ), &StatusMsg,
                                          pContext-
>TxnContext.ReducerID,
                                          pContext-
>TxnContext.ReducerID );


  PRTEpause( );

  LoginMsg.Type = USER_LOGIN_REPLY_MSG;


  if( SCR_E_SUCCESS != ( LoginMsg.Status = Init( pContext )))
  {

    PRTEmessage_to_user_binary( sizeof( LoginMsg ), &LoginMsg,
                                                pContext-
>TxnContext.ReducerID,
                                                pContext-
>TxnContext.ReducerID );
    DisconnectExit ( pContext, LoginMsg.Status, FALSE );
  }

  if( SCR_E_SUCCESS != ( LoginMsg.Status = StartupTxn( &pContext-
>TxnContext )))
  {

    PRTEmessage_to_user_binary( sizeof( LoginMsg ), &LoginMsg,
                                                pContext-
>TxnContext.ReducerID,
                                                pContext-
>TxnContext.ReducerID );
    DisconnectExit( pContext, LoginMsg.Status, FALSE );
  }


  LoginMsg.Status = ConnectTxn( &pContext->TxnContext,
CONNECT_TRANS,
                                                &LoginMsg.Data );
  if( SCR_E_SUCCESS != LoginMsg.Status )
  {

    PRTEmessage_to_user_binary( sizeof( LoginMsg ), &LoginMsg,
                                                pContext-
>TxnContext.ReducerID,
                                                pContext-
>TxnContext.ReducerID );
    DisconnectExit( pContext, LoginMsg.Status, FALSE );
  }


  LoginMsg.Status = SCR_E_SUCCESS;

  PRTEmessage_to_user_binary( sizeof(LoginMsg), &LoginMsg,
                                          pContext-
>TxnContext.ReducerID,
                                          pContext-
>TxnContext.ReducerID );


  StatusMsg.Status = DequeueExecLoopTxn( &pContext->TxnContext );



  DisconnectExit( pContext, StatusMsg.Status, TRUE );
}


void
BinMsgHandler( void *Context, int SendersID, int Len, void *Msg )
{
  user_context_t          *pContext;



  generic_msg_t                   *pMsg;
  start_msg_t                     *pStartMsg;
  txn_msg_t               *pTxnMsg;


  pMsg = Msg;
  pContext = Context;


  switch( pMsg->Type ) {
    case USER_LOGIN_REQUEST_MSG:
      PRTEresume( );
      break;

    case USER_START_REQUEST_MSG:
      pStartMsg = Msg;
      EnqueueTxn( &pContext->TxnContext, TXN_CACHE_SIZE,
&pStartMsg->Data[0] );
      break;

    case TXN_MSG:
      pTxnMsg = Msg;
      EnqueueTxn( &pContext->TxnContext, 1, &pTxnMsg->Data );
      break;

    case USER_STOP_REQUEST_MSG:
      PRTEstop( );
      PRTEdelay( 0.0 );
      break;

    default:
      SCRlog( "User: Received unknown message type: %d", *pMsg );
  }
}


void
DisconnectExit(user_context_t *pContext, int Status, int Disconnect)
{
  user_exit_msg_t         ExitMsg;
  char                            tmp_buf[128];
  int                             LocStatus;


  if ( TRUE == Disconnect )
  {
    LocStatus = DisconnectTxn( &pContext->TxnContext,
DISCONNECT_TRANS );
    if( SCR_E_SUCCESS != LocStatus )
    {
      SCRerr( LocStatus );
      SCRlog( "User %d: Failed to disconnect from application.", User_id );
```

```
  }
 }

 LocStatus = ShutdownTxn( &pContext->TxnContext );
 if( SCR_E_SUCCESS != LocStatus )
 {
  SCRerr( LocStatus );
  SCRlog( "User %d: Failed to shutdown application.", User_id );
 }



 Flush_log = TRUE;
 if (Status == SCR_E_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 );
  SCRerr( 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-
>TxnContext.ReducerID,
                                                pContext-
>TxnContext.ReducerID );

 PRTEexit();
}


int
Init( user_context_t *pContext )
{
 char temp_string[128];
 char *pNetVarStr;
 char *master_seed;
 int  temp_int;


 SCR_getnet_string("LOG_DIR", LOG_DIR_DEFAULT, temp_string);
 if ('\0' == temp_string[0] )
 {
  return( SCRerr( SCR_E_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( SCR_E_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( SCR_E_CODE_VERSION_MISMATCH ));
 }



 SCR_getnet_int("TPCC_USER_FLUSH_LOG",
TPCC_USER_FLUSH_LOG_DEFAULT,
                          &Flush_log);


 SCR_getnet_int( "RUN_MODE", RUN_MODE_DEFAULT,
                           &pContext->TxnContext.RunMode );


 SCR_getnet_string( "CGI_SCRIPT_NAME",
CGI_SCRIPT_NAME_DEFAULT,
                            pContext->TxnContext.CgiScriptName );
 if( '\0' == *pContext->TxnContext.CgiScriptName )
 {
  return( SCRerr( SCR_E_CGI_SCRIPT_NAME_NOT_SET ));
 }



 master_seed = PRTEwait_for_network_variable("SEED");
 pContext->myseed = ( atol( master_seed ) * User_id * 2000 ) + 1;

 PRTErandomize(((unsigned) pContext->myseed % 100));
 srand48(pContext->myseed);


 SCR_getnet_bool( "DURABILITY_LOGGING",
DURABILITY_LOGGING_DEFAULT,
                          &pContext->TxnContext.DurabilityLogging );


 pNetVarStr = PRTEget_network_variable( "FIRST_USER_ID" );
 if( NULL == pNetVarStr )
 {
  return( SCRerr( SCR_E_FIRST_USER_ID_NOT_SET ));
 }

 temp_int = User_id - atoi( pNetVarStr );
 pContext->TxnContext.w_id = ( temp_int / 10 ) + 1;
 pContext->TxnContext.d_id = ( temp_int % 10 ) + 1;


 SCR_getnet_int( "C_LAST", C_LAST_DEFAULT, &( pContext-
>TxnContext.CLast ));

 return( SCR_E_SUCCESS );
}
```

Appendix D

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
        memory alignment boundary = DEFAULT
        global async prefetch limit = 0
        global cache partition number = DEFAULT

[Named Cache:c_customer]
        cache size = 4335M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 4335M
        wash size = 4096 K
        local async prefetch limit = 0

[Named Cache:c_customer_index]
        cache size = 500M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 500M
        wash size = 2048 K
        local async prefetch limit = 0

[Named Cache:c_log]
        cache size = 30M
        cache status = log only
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 1

[4K I/O Buffer Pool]
        pool size = 10M
        wash size = 512 K
        local async prefetch limit = 0

[8K I/O Buffer Pool]
        pool size = 20M
        wash size = 2048 K
        local async prefetch limit = 0

[Named Cache:c_no]
        cache size = 810M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 810M
        wash size = 64 K
        local async prefetch limit = 0

[Named Cache:c_non_customer_index]
        cache size = 1050M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 1050M
        wash size = 2048 K
        local async prefetch limit = 0

[Named Cache:c_ol]
        cache size = 700M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 700M
        wash size = 256 K
        local async prefetch limit = 0

[Named Cache:c_ol_index]
        cache size = 380M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 380M
        wash size = 2048 K
        local async prefetch limit = 0

[Named Cache:c_orders]
        cache size = 1100M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 775M
        wash size = 2048 K
        local async prefetch limit = 0

[16K I/O Buffer Pool]
        pool size = 325M
        wash size = 2048 K
        local async prefetch limit = 0

[Named Cache:c_stock]
        cache size = 20000M
        cache status = mixed cache
        cache replacement policy = DEFAULT
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 20000M
        wash size = 3144000 K
        local async prefetch limit = 0

[Named Cache:c_stock_index]
        cache size = 500M
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 500M
        wash size = 512 K
        local async prefetch limit = 0

[Named Cache:c_tinyhot]
        cache size = 25M
```

```
        cache status = mixed cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 25M
        wash size = 64 K
        local async prefetch limit = 0

[Named Cache:default data cache]
        cache size = 140M
        cache status = default data cache
        cache replacement policy = relaxed LRU replacement
        local cache partition number = 4

[4K I/O Buffer Pool]
        pool size = 140M
        wash size = 256 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
        partition groups = DEFAULT
        partition spinlock ratio = DEFAULT

[Disk I/O]
        disk i/o structures = DEFAULT
        number of large i/o buffers = DEFAULT
        page utilization percent = DEFAULT
        number of devices = 150
        disable disk mirroring = DEFAULT
        allow sql server async i/o = DEFAULT

[Languages]
        disable character set conversions = DEFAULT

[Unicode]
        enable unicode normalization = DEFAULT
        enable surrogate processing = 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
        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
        allow remote access = DEFAULT

[O/S Resources]
        max async i/os per engine = 716
        max async i/os per server = 716

[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]
        max memory = 16406621
        additional network memory = 4915200
        shared memory starting address = DEFAULT
        allocate max shared memory = 1
        dynamic allocation on demand = DEFAULT
        lock shared memory = DEFAULT
        heap memory per user = DEFAULT

[Processors]
        max online engines = 4
        number of engines at startup = 4

[SQL Server Administration]
        procedure cache size = 204800
        default database size = DEFAULT
        identity burning set factor = DEFAULT
        allow nested triggers = DEFAULT
        allow updates to system tables = 1
        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
        cpu grace time = DEFAULT
        number of sort buffers = DEFAULT
        size of auto identity column = DEFAULT
        identity grab size = DEFAULT
        housekeeper free write percent = 0
        enable housekeeper GC = 0
        allow resource limits = DEFAULT
        number of aux scan descriptors = DEFAULT
        SQL Perfmon Integration = DEFAULT
        allow backward scans = DEFAULT
        license information = DEFAULT
        enable sort-merge join and JTC = DEFAULT
        abstract plan load = DEFAULT
        abstract plan dump = DEFAULT
        abstract plan replace = DEFAULT
        abstract plan cache = DEFAULT
        text prefetch size = DEFAULT
        enable HA = DEFAULT

[User Environment]
        number of user connections = 250
        stack size = DEFAULT
        stack guard size = DEFAULT
        permission cache entries = DEFAULT
        user log cache size = DEFAULT
        user log cache spinlock ratio = DEFAULT

[Lock Manager]
        number of locks = DEFAULT
        deadlock checking period = DEFAULT
        lock spinlock ratio = DEFAULT
        lock address spinlock ratio = DEFAULT
        lock table spinlock ratio = DEFAULT
        lock hashtable size = DEFAULT
        lock scheme = DEFAULT
        lock wait period = DEFAULT
        read committed with lock = DEFAULT
        print deadlock information = DEFAULT
        deadlock retries = DEFAULT
        page lock promotion HWM = DEFAULT
        page lock promotion LWM = DEFAULT
        page lock promotion PCT = DEFAULT
        row lock promotion HWM = DEFAULT
        row lock promotion LWM = DEFAULT
        row lock promotion PCT = 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
        enable row level access = DEFAULT
```

```
        check password for digit = DEFAULT
        minimum password length = DEFAULT
        maximum failed logins = DEFAULT
        enable ssl = DEFAULT
        unified login required = DEFAULT
        use security services = DEFAULT
        msg confidentiality reqd = DEFAULT
        msg integrity reqd = DEFAULT
        secure default login = 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

[Component Integration Services]
        enable cis = 0
        cis connect timeout = DEFAULT
        cis bulk insert batch size = DEFAULT
        max cis remote connections = DEFAULT
        cis packet size = DEFAULT
        cis cursor rows = DEFAULT
        enable file access = DEFAULT
        cis bulk insert array size = DEFAULT
        enable full-text search = DEFAULT
        cis rpc handling = DEFAULT

[Java Services]
        enable java = DEFAULT
        size of process object heap = DEFAULT
        size of shared class heap = DEFAULT
        size of global fixed heap = DEFAULT
        number of java sockets = DEFAULT
        enable enterprise java beans = DEFAULT

[DTM Administration]
        enable DTM = DEFAULT
        enable xact coordination = 0
        xact coordination interval = DEFAULT
        number of dtx participants = DEFAULT
        strict dtm enforcement = DEFAULT
        txn to pss ratio = DEFAULT
        dtm lock timeout period = DEFAULT
        dtm detach timeout period = DEFAULT

[Diagnostics]
        dump on conditions = DEFAULT
        maximum dump conditions = DEFAULT
        number of ccbs = DEFAULT
        caps per ccb = DEFAULT
        average cap size = DEFAULT

[Monitoring]
        enable monitoring = DEFAULT
        sql text pipe active = DEFAULT
        sql text pipe max messages = DEFAULT
        plan text pipe active = DEFAULT
        plan text pipe max messages = DEFAULT
        statement pipe active = DEFAULT
        statement pipe max messages = DEFAULT
        errorlog pipe active = DEFAULT
        errorlog pipe max messages = DEFAULT
        deadlock pipe active = DEFAULT
        deadlock pipe max messages = DEFAULT
        wait event timing = DEFAULT
        process wait events = DEFAULT
        object lockwait timing = DEFAULT
        SQL batch capture = DEFAULT
        statement statistics active = DEFAULT
        per object statistics active = DEFAULT
        max SQL text monitored = DEFAULT
```

database_init_script

```
#!/bin/sh -f

# Set configuration parameters for checkpoint on large memory.

isql -Usa -P <<- EOF

use tpcc
go

dbcc tune(maxwritedes, 10)
go

dbcc tune("doneinproc", 0)
go

dbcc tune("cleanup", 0)
go


dbcc tune(freelock_xfr_bsize, 150)
go
dbcc tune(max_eng_freelocks, 50)
go

dbcc tune(des_bind, 4, warehouse)
dbcc tune(des_bind, 4, district)
dbcc tune(des_bind, 4, item)
dbcc tune(des_bind, 4, stock)
dbcc tune(des_bind, 4, order_line)
dbcc tune(des_bind, 4, orders)
dbcc tune(des_bind, 4, new_order)
dbcc tune(des_bind, 4, customer)
dbcc tune(des_bind, 4, history)
dbcc tune(des_bind, 4, neworder_local)
dbcc tune(des_bind, 4, neworder_remote)
dbcc tune(des_bind, 4, payment_byid)
dbcc tune(des_bind, 4, payment_byname)
dbcc tune(des_bind, 4, order_status_byid)
dbcc tune(des_bind, 4, order_status_byname)
dbcc tune(des_bind, 4, delivery)
dbcc tune(des_bind, 4, stock_level)
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 traceoff(-1)
go

EOF
```

kernel_config

```
ident                   "PRIVATE1"

options                 AUTOFS
options                 BIN_COMPAT
options                 BSD_TTY
options                 BUFCACHE_STATS
options                 COMPAT_43
options                 DLB
options                 DLI
options                 FFM_FS
options                 INET
options                 INOCACHE_STATS
options                 LABELS
options                 LDTTY
options                 MACH
options                 MACH_IPC_TCACHE
options                 MACH_IPC_WWA
options                 MACH_IPC_XXXHACK
options                 NFS
options                 NFS_SERVER
options                 OSF
options                 QUOTA
options                 RPTY
options                 STAT_TIME
options                 STREAMS
options                 STRKINFO
options                 SYSV_COFF
options                 UERF
options                 UFS
options                 UIPC
options                 VAGUE_STATS
options                 _LMF_
options     BPARM
options     PROCFS
options     SL
options SNMPINFO

#
# 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
#
makeoptionsCCOMPRESS="-compress"
makeoptionsCDEBUGOPTS="-g3"
makeoptionsPROFOPTS="-DPROFILING -DPROFTYPE=4"
makeoptionsLOADADDR="fffffc0000630000"
#
# Special options (see configuring the kernel chapter
# in the Guide to System Administration)
#
cpu                     "ALPHATITAN"
machine                 alpha

config                  vmunix      swap generic

bus         isa0    at *
callout after_c "../bin/mkdata isa"


#
# Static Driver Definitions
#
config_driver alt
config_driver ata
config_driver scsi
config_driver ciss
config_driver fdi
config_driver lp
config_driver ace
config_driver gpc
config_driver pci
callout after_c "../bin/mkdata pci"

#
# Pseudodevice Definitions (see configuring the
# kernel chapter in the Guide to System Administration)
#
pseudo-device           ether
pseudo-device           loop
pseudo-device           lsm         1
pseudo-device           lsm_vmted   0
pseudo-device           rt_hab
pseudo-device           soe_two_hab
pseudo-device           svid_three_hab
pseudo-device           svr_four_hab
pseudo-device           sysv_hab
pseudo-device           ws
```

sysconfigtab

```
#
#
************************************************************
****
# *                                                      *
# *   Copyright 2002 Compaq Information Technologies Group, L.P.  *
# *                                                      *
# *   The software contained on this media  is  proprietary  to   *
# *   and  embodies  the  confidential  technology  of  Compaq    *
# *   Computer Corporation.  Possession, use,  duplication  or    *
# *   dissemination of the software and media is authorized only  *
# *   pursuant to a valid written license from Compaq Computer    *
# *   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

#
```

```
#
# The supported method of changing the information in this
# file (sysconfigtab) is by using the sysconfigdb command.
#
# If you insist upon changing this file by direct editing,
# then it would be advisable to add new information at the end
# rather than the beginning or middle of the file, in order
# to reduce confusion during installation and reconciliation
# of user changes and update changes.
#
# See the appropriate documentation such as the release notes
# for the new version being installed as well as the appropriate
# reference pages.
#

#
# Turn on CAM parallel probing
#
io:
                parallel_edt_scan = 0
                basic_dma_window_size = 2048
                basic_dma_window_size = 2048

#
#  Contiguous Memory Allocation for loadable driver subsystems
#
cma_dd:



#
#  EISA_Option = Board_Id Function_Name Driver_Name Type
#
eisa:
#
# %%%EISA
#
  EISA_Option = Board_Id - ADP0001, Function_Name - AHA1740,
Driver_Name - aha, Int_Aft_Attach - 1
  EISA_Option = Board_Id - ADP0002, Function_Name - AHA1740,
Driver_Name - aha, Int_Aft_Attach - 1
  EISA_Option = Board_Id - DEC2500, Driver_Name - envram
  EISA_Option = Board_Id - ISA1010, Function_Name  - 'COM,1'
Driver_Name - ace
  EISA_Option = Board_Id - ISA1010, Function_Name - 'COM,2',
Driver_Name - ace
  EISA_Option = Board_Id - ISA1010, Function_Name - PAR,
Driver_Name - lp
  EISA_Option = Board_Id - DEC2A01, Function_Name - SYSMEM,
Int_Aft_Attach - 1
  EISA_Option = Board_Id - DEC2A01, Function_Name - 'ACECOM,1',
Driver_Name - ace
  EISA_Option = Board_Id - DEC2A01, Function_Name - 'ACECOM,2',
Driver_Name - ace
  EISA_Option = Board_Id - DEC2A01, Function_Name - PAR,
Driver_Name - lp
  EISA_Option = Board_Id - MLX0070, Driver_Name - xcr,
Int_Aft_Attach - 1
  EISA_Option = Board_Id - MLX0075, Driver_Name - xcr,
Int_Aft_Attach - 1
  EISA_Option = Board_Id - MLX0077, Driver_Name - xcr,
Int_Aft_Attach - 1
  EISA_Option = Board_Id - DEC5000, Function_Name - SYSMEM,
Int_Aft_Attach - 1
  EISA_Option = Board_Id - DEC5000, Function_Name - 'ACECOM,1',
Driver_Name - ace
  EISA_Option = Board_Id - DEC5000, Function_Name - 'ACECOM,2',
Driver_Name - ace
  EISA_Option = Board_Id - DEC5000, Function_Name - PAR,
Driver_Name - lp
  EISA_Option = Board_Id - DEC5000, Function_Name - 'KBD,MOUSE',
Driver_Name - gpc
  EISA_Option = Board_Id - DEC5100, Function_Name - SYSMEM,
Int_Aft_Attach - 1
  EISA_Option = Board_Id - DEC5100, Function_Name - 'ACECOM,1',
Driver_Name - ace
  EISA_Option = Board_Id - DEC5100, Function_Name - 'ACECOM,2',
Driver_Name - ace
  EISA_Option = Board_Id - DEC5100, Function_Name - PAR,
Driver_Name - lp
  EISA_Option = Board_Id - DEC5100, Function_Name - 'KBD,MOUSE',
Driver_Name - gpc
  EISA_Option = Board_Id - DEC5301, Function_Name - SYSMEM,
Int_Aft_Attach - 1
  EISA_Option = Board_Id - DEC5301, Function_Name - 'ACECOM,1',
Driver_Name - ace
  EISA_Option = Board_Id - DEC5301, Function_Name - 'ACECOM,2',
Driver_Name - ace
  EISA_Option = Board_Id - DEC5301, Function_Name - PAR,
Driver_Name - lp
  EISA_Option = Board_Id - DEC5301, Function_Name - 'KBD,MOUSE',
Driver_Name - gpc
  EISA_Option = Board_Id - DEC6000, Function_Name - 'SYSMEM',
Int_Aft_Attach - 1
  EISA_Option = Board_Id - DEC6000, Function_Name - 'KBD,MOUSE',
Driver_Name - gpc
  EISA_Option = Board_Id - DEC6400, Function_Name - SYSMEM,
Driver_Name - Null
  EISA_Option = Board_Id - DEC6400, Function_Name - 'ACECOM,1',
Driver_Name - ace
  EISA_Option = Board_Id - DEC6400, Function_Name - 'ACECOM,2',
Driver_Name - ace
  EISA_Option = Board_Id - DEC6400, Function_Name - PAR,
Driver_Name - lp
  EISA_Option = Board_Id - DEC6400, Function_Name - 'KBD,MOUSE',
Driver_Name - gpc
#
#  ISA_Option = Board_Id Function_Name Driver_Name Type
#
isa:
#
# %%%ISA
#
  ISA_Option =  Function_Name - 'KBD,MOUSE', Driver_Name - gpc
  ISA_Option =  Function_Name - COM, Driver_Name - ace
  ISA_Option =  Function_Name - LPT, Driver_Name - lp
  ISA_Option =  Function_Name - 'ISA--SCC', Driver_Name - iscc



#
# Pcisw_Version Vendor_Id Ddevice_Id Rev Base Sub Pif Sub_Vid
Sub_Did Vid_Mo_Flag Did_Mo_Flag
# Rev_Mo_Flag Base_Mo_Flag Sub_Mo_Flag Pif_Mo_Flag
Sub_Vid_Mo_Flag Driver_Name Type
#
pci:
#
# %%%PCI
#

#
#  PCI to something bridge adapters
#
#  At present these are not used in system configuration, but are present in
#  this file for informational purposes.  In some cases, the "driver" may
#  not exist, and in other cases the driver may exist but be unprepared to
#  parse these entries, so they are all commented out at this time.

#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1013, Device_Id -
0x1100, Driver_Name - pcmcia, Type - A, Adpt_Config - N
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Device_Id -
0x1, Base - 6, Sub - 4, Driver_Name - pci, Type - A, Adpt_Config - N
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Base - 6, Sub
- 4, Driver_Name - pci, Type - A, Adpt_Config - N
#  PCI_Option =  PCI_SE_Rev - 0x210, Base - 6, Sub - 4, Driver_Name -
pci, Type - A, Adpt_Config - N
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x8086, Device_Id -
0x482, Driver_Name - eisa, Type - A, Adpt_Config - N
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x8086, Device_Id -
0x484, Driver_Name - isa, Type - A, Adpt_Config - N
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Device_Id -
0x10, Driver_Name - vba, Type - A, Adpt_Config - N
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x10E3, Device_Id -
0x00, Driver_Name - vba, Type - A, Adpt_Config - N



#
#  TC_Option = Modname Driver_Name
```

```
#
tc:
#
# %%%TC
#
   TC_Option =  Modname - 'PMTNV-AA', Driver_Name - nvtc,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAP-AA ', Driver_Name - nvtc,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAZ-AA ', Driver_Name - asc,
Int_Aft_Attach - 1, Type - A, Adpt_Config - N
   TC_Option =  Modname - 'PMAZ-DS ', Driver_Name - tcds,
Int_Aft_Attach - 1, Type - A, Adpt_Config - N
   TC_Option =  Modname - 'PMAZ-FS ', Driver_Name - tcds,
Int_Aft_Attach - 1, Type - A, Adpt_Config - N
   TC_Option =  Modname - 'PMAZB-AA', Driver_Name - tcds,
Int_Aft_Attach - 1, Type - A, Adpt_Config - N
   TC_Option =  Modname - 'PMAZB-AB', Driver_Name - tcds,
Int_Aft_Attach - 1, Type - A, Adpt_Config - N
   TC_Option =  Modname - 'PMAZC-AA', Driver_Name - tcds,
Int_Aft_Attach - 1, Type - A, Adpt_Config - N
   TC_Option =  Modname - 'KZTSA-AA', Driver_Name - tza,
Int_Aft_Attach - 1, Type - A, Adpt_Config - N
   TC_Option =  Modname - 'PMAGB-BA', Driver_Name - fb,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAG-RO ', Driver_Name - fb, Type - C,
Adpt_Config - N
   TC_Option =  Modname - 'PMAG-JA ', Driver_Name - fb, Type - C,
Adpt_Config - N
   TC_Option =  Modname - 'PMADC   ', Driver_Name - pv, Int_Aft_Attach
- 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAGC-AA', Driver_Name - pv,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAGC-BA', Driver_Name - pv,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAGC-DA', Driver_Name - pvl,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAGC-EA', Driver_Name - pvl,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAGD   ', Driver_Name - fb, Int_Aft_Attach
- 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAGD-AA', Driver_Name - fb,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
   TC_Option =  Modname - 'PMAGD-BA', Driver_Name - fb,
Int_Aft_Attach - 1, Type - C, Adpt_Config - N
# Disabled due to lack of X support in Platinum, device driver is being
pulled from the release
#   TC_Option =  Modname - "KWS_TD", Confname - kws_td,
Int_Aft_Attach- 1, Type - C, Adpt_Config - N

#
# fdi: FLOPPY
#
fdi:
   ISA_Option =  Function_Name - FLOPPY, Driver_Name - fdi
   EISA_Option = Board_Id - ADP0002, Function_Name - 'MSD,FPYCTL',
Driver_Name - fdi
   EISA_Option = Board_Id - DEC2400, Function_Name - 'MSD,FPYCTL',
Driver_Name - fdi
   EISA_Option = Board_Id - DEC2A01, Function_Name - 'MSD,FPYCTL',
Driver_Name - fdi
   EISA_Option = Board_Id - DEC5000, Function_Name - 'MSD,FPYCTL',
Driver_Name - fdi
   EISA_Option = Board_Id - DEC5100, Function_Name - 'MSD,FPYCTL',
Driver_Name - fdi
   EISA_Option = Board_Id - DEC5301, Function_Name - 'MSD,FPYCTL',
Driver_Name - fdi
   EISA_Option = Board_Id - DEC6000, Function_Name - 'MSD,FPYCTL',
Driver_Name - fdi

#
# ln: LANCE Ethernet
#
ln:
   TC_Option =  Modname - 'PMAD-AA ', Driver_Name - ln, Type - C,
Adpt_Config - N
   TC_Option =  Modname - 'PMAD-BA ', Driver_Name - ln, Type - C,
Adpt_Config - N
   EISA_Option = Board_Id - DEC4220, Function_Name - 'NET,ETH',
Driver_Name - ln
   CMA_Option = Size - 0x10000, Alignment - 0x10000, Addrlimit - 0,
Type - 0x1f, Flag - 0

#
# le: LeMAC Ethernet
#
le:
   ISA_Option =  Function_Name - 'DE200-LE', Driver_Name - le

#
# el: 3Com EtherLink III Ethernet
#
el:
   PCMCIA_Option = Manufact_Name - '3Com Corporation',
Product_Name - '3C589', Manufact_Id - 0x101, Card_Rev - 0, Func_Code -
6, Driver_Name - el, Loadable_Flag - 0, Unload_Flag - 0,
Intr_Handler_Flag - 0, Type - C, Adpt_Config - N, Man_Name_Mo_Flag -
1, Prd_Name_Mo_Flag - 1, Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0,
Multi_Func_Flag - 0, Func_Num - 0

   PCMCIA_Option = Manufact_Name - '3Com Corporation',
Product_Name - '3C589D', Manufact_Id - 0x101, Card_Rev - 0, Func_Code
- 6, Driver_Name - el, Loadable_Flag - 0, Unload_Flag - 0,
Intr_Handler_Flag - 0, Type - C, Adpt_Config - N, Man_Name_Mo_Flag -
1, Prd_Name_Mo_Flag - 1, Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0,
Multi_Func_Flag - 0, Func_Num - 0

   ISA_Option =  Function_Name - 3C509, Driver_Name - el, Type - C

   PCMCIA_Option = Manufact_Name - '3Com Corporation',
Product_Name - '3C562B/3C563B', Manufact_Id - 0x101, Card_Rev - 0,
Func_Code - 6, Driver_Name - el, Loadable_Flag - 0, Unload_Flag - 0,
Intr_Handler_Flag - 0, Type - C, Adpt_Config - N, Man_Name_Mo_Flag -
1, Prd_Name_Mo_Flag - 1, Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0,
Multi_Func_Flag - 1, Func_Num - 0

#
# ee: i82558/9 PCI 10/100 Ethernet
#
ee:
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x8086, Device_Id -
0x1229, Driver_Name - ee

#
# tu: TULIP Ethernet/Fast Ethernet
#
tu:
   EISA_Option = Board_Id - DEC4250, Function_Name - Null,
Driver_Name - tu,  Type - C, Adpt_Config - N
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Device_Id -
2, Driver_Name - tu
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Device_Id -
0x14, Driver_Name - tu
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Device_Id -
0x9, Driver_Name - tu
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Device_Id -
0x19, Driver_Name - tu

#
# alt: DEGPA Gigabit Ethernet
#
alt:
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x12AE, Device_Id -
0x1, Driver_Name - alt

#
# fza: DEFZA FDDI
#
fza:
   TC_Option =  Modname - 'PMAF-AA ', Driver_Name - fza, Type - C,
Adpt_Config - N

#
# fta: PDQ FDDI
#
fta:
   TC_Option =  Modname - 'PMAF-FA ', Driver_Name - fta, Type - C,
Adpt_Config - N
   TC_Option =  Modname - 'PMAF-FS ', Driver_Name - fta, Type - C,
Adpt_Config - N
```

```
  TC_Option =  Modname - 'PMAF-FD ', Driver_Name - fta, Type - C,
Adpt_Config - N
  TC_Option =  Modname - 'PMAF-FU ', Driver_Name - fta, Type - C,
Adpt_Config - N
  EISA_Option = Board_Id - DEC3001, Driver_Name - fta
  EISA_Option = Board_Id - DEC3002, Driver_Name - fta
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Device_Id -
0xF, Driver_Name - fta

#
# tra: Token-Ring
#
tra:
  TC_Option =  Modname - 'PMAT-AA ', Driver_Name - tra, Type - C,
Adpt_Config - N
  ISA_Option =  Function_Name - DW110, Driver_Name - tra
  ISA_Option =  Function_Name - TKRNG, Driver_Name - tra
  EISA_Option = Board_Id - PRO6000, Driver_Name - tra
  EISA_Option = Board_Id - PRO6001, Driver_Name - tra
  EISA_Option = Board_Id - PRO6002, Driver_Name - tra
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x104C, Device_Id -
0x0508, Driver_Name - tra
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x10DA, Device_Id -
0x0508, Driver_Name - tra
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x10EF, Device_Id -
0x8154, Driver_Name - tra

#
# lta: ATM
#
# NOTE:  There must be one CMA_Option line for each dynamically
#       loaded Turbo Channel lta device (DGLTA-FA)
#
lta:
  TC_Option =  Modname - 'DGLTA-FA',Driver_Name - lta, Type - C,
Adpt_Config - N
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1011, Device_Id -
0x16, Driver_Name - lta
#  CMA_Option = Size - 0x23D00, Alignment - 0x10000, Addrlimit - 0,
Type - 0x1d, Flag - 2

#
# lfa:  ForeRunner HE Series ATM Adapter
#
lfa:
#  PCI_Option = PCI_SE_Rev - 0x210, Vendor_Id - 0x1127, Device_Id -
0x400, Rev - 0, Base - 0, Sub - 0, Pif - 0, Sub_Vid - 0, Sub_Did - 0,
Vid_Mo_Flag - 1, Did_Mo_Flag - 1, Rev_Mo_Flag - 0, Base_Mo_Flag - 0,
Sub_Mo_Flag - 0, Pif_Mo_Flag - 0, Sub_Vid_Mo_Flag - 0,
Sub_Did_Mo_Flag - 0, Driver_Name - lfa, Type - C, Adpt_Config - N

#
# PCMCIA_Option = Manufact_Name Product_Name Manufact_Id
Card_Rev Func_Code
#                       Driver_Name Loadable_Flag Unload_Flag
Intr_Handler_Flag Type
#                               Adpt_Config Man_Name_Mo_Flag
Prd_Name_Mo_Flag Mid_Mo_Flag Card_Rev_Mo_Flag
#                               Multi_Func_Flag Func_Num
#
pcmcia:
#
# %%%PCMCIA
#
  PCMCIA_Option = Manufact_Name - 'AT&T Paradyne', Product_Name -
'KeepInTouch Card', Manufact_Id - 0, Card_Rev - 0, Func_Code - 2,
Driver_Name - ace, Loadable_Flag - 0, Unload_Flag - 0, Intr_Handler_Flag
- 1, Type - C, Adpt_Config - N, Man_Name_Mo_Flag - 1,
Prd_Name_Mo_Flag - 1, Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0,
Multi_Func_Flag - 0, Func_Num - 0
  PCMCIA_Option = Manufact_Name - 'Digital', Product_Name -
'PCMCIA V.32bis 14,400 Fax', Manufact_Id - 0, Card_Rev - 0, Func_Code
- 2, Driver_Name - ace, Loadable_Flag - 0, Unload_Flag - 0,
Intr_Handler_Flag - 1, Type - C, Adpt_Config - N, Man_Name_Mo_Flag -
1, Prd_Name_Mo_Flag - 1, Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0,
Multi_Func_Flag - 0, Func_Num - 0
  PCMCIA_Option = Manufact_Name - 'MEGAHERTZ', Product_Name -
'XJ2288', Manufact_Id - 0, Card_Rev - 0, Func_Code - 2, Driver_Name -
ace, Loadable_Flag - 0, Unload_Flag - 0, Intr_Handler_Flag - 1, Type - C,
Adpt_Config - N, Man_Name_Mo_Flag - 1, Prd_Name_Mo_Flag - 1,
Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0, Multi_Func_Flag - 0,
Func_Num - 0
  PCMCIA_Option = Manufact_Name - '3Com Corporation',
Product_Name - '3C562B/3C563B', Manufact_Id - 0x101, Card_Rev - 0,
Func_Code - 2, Driver_Name - ace, Loadable_Flag - 0, Unload_Flag - 0,
Intr_Handler_Flag - 1, Type - C, Adpt_Config - N, Man_Name_Mo_Flag -
1, Prd_Name_Mo_Flag - 1, Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0,
Multi_Func_Flag - 1, Func_Num - 1
  PCMCIA_Option = Manufact_Name - 'AD PC-CARD', Product_Name -
'RC288ACL', Manufact_Id - 0, Card_Rev - 0, Func_Code - 2, Driver_Name
- ace, Loadable_Flag - 0, Unload_Flag - 0, Intr_Handler_Flag - 1, Type - C,
Adpt_Config - N, Man_Name_Mo_Flag - 1, Prd_Name_Mo_Flag - 1,
Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0, Multi_Func_Flag - 0,
Func_Num - 0
  PCMCIA_Option = Manufact_Name - 'AD2880WRLD', Product_Name -
'International_V.34_PC-Card_Modem', Manufact_Id - 0, Card_Rev - 0,
Func_Code - 2, Driver_Name - ace, Loadable_Flag - 0, Unload_Flag - 0,
Intr_Handler_Flag - 1, Type - C, Adpt_Config - N, Man_Name_Mo_Flag -
1, Prd_Name_Mo_Flag - 1, Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0,
Multi_Func_Flag - 0, Func_Num - 0
#
# The following are the ONLY ATA cards that have been tested.  Others are
known
# to violate various parts of the spec and therefore fail.  Some devices also
# require ddr.dbase entries due to broken firmware which causes them to
self-
# destruct (the Maxtor DDR entry is an example).  Beware if you attempt
using
# such devices.
#
  PCMCIA_Option = Manufact_Name - 'INTEGRAL PERIPHERALS',
Product_Name - 'ATA CARD', Manufact_Id - 0, Card_Rev - 0, Func_Code -
4, Driver_Name - ata, Loadable_Flag - 0, Unload_Flag - 0,
Intr_Handler_Flag - 0, Type - A, Adpt_Config - N, Man_Name_Mo_Flag -
1, Prd_Name_Mo_Flag - 1, Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0,
Multi_Func_Flag - 0, Func_Num - 0
  PCMCIA_Option = Manufact_Name - 'IBM', Product_Name - ",
Manufact_Id - 0, Card_Rev - 0, Func_Code - 4, Driver_Name - ata,
Loadable_Flag - 0, Unload_Flag - 0, Intr_Handler_Flag - 0, Type - A,
Adpt_Config - N, Man_Name_Mo_Flag - 1, Prd_Name_Mo_Flag - 0,
Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0, Multi_Func_Flag - 0,
Func_Num - 0
  PCMCIA_Option = Manufact_Name - 'Maxtor', Product_Name - ",
Manufact_Id - 0, Card_Rev - 0, Func_Code - 4, Driver_Name - ata,
Loadable_Flag - 0, Unload_Flag - 0, Intr_Handler_Flag - 0, Type - A,
Adpt_Config - N, Man_Name_Mo_Flag - 1, Prd_Name_Mo_Flag - 0,
Mid_Mo_Flag - 0, Card_Rev_Mo_Flag - 0, Multi_Func_Flag - 0,
Func_Num - 0


#
#  qvision: CMPQ Qvision SVGA
#
qvision:
  EISA_Option = Board_Id - CPQ3011, Driver_Name - qvision
  EISA_Option = Board_Id - CPQ3111, Driver_Name - qvision
  EISA_Option = Board_Id - CPQ3112, Driver_Name - qvision
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x0e11, Device_Id -
0x3032, Driver_Name - qvision

#
#  cirrus: Cirrus SVGA
#
cirrus:
  EISA_Option = Board_Id - DEC5000, Function_Name - VID,
Driver_Name - cirrus

#
#  ati64:  ATI Mach64 SVGA
#
ati64:
  EISA_Option = Board_Id - ISA6400, Driver_Name - ati64_vga
  ISA_Option =  Function_Name - 'MACH64', Driver_Name - ati64_vga
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1002, Device_Id -
0x4358, Driver_Name - ati
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1002, Device_Id -
0x4758, Driver_Name - ati
#  PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1002, Device_Id -
0x4354, Driver_Name - ati
```

```
#
#  s3trio:  S3 Trio 32/64 SVGA
#
s3trio:
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x5333, Device_Id -
0x8811, Driver_Name - trio
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x5333, Device_Id -
0x8810, Driver_Name - trio

#
#  wd:  Western Digital WD90C24 SVGA
#
   ISA_Option =  Function_Name - 'WD90', Driver_Name - wd

#
#  vga: Standard 640x480 VGA
#
vga:
   EISA_Option = Board_Id - PHI8041, Driver_Name - vga
   ISA_Option =  Function_Name - 'ISA--VGA', Driver_Name - vga
#   PCI_Option =  PCI_SE_Rev - 0x210, Base - 3, Sub - 0, Driver_Name -
vga
#   PCI_Option =  PCI_SE_Rev - 0x210, Base - 0, Sub - 1, Driver_Name -
vga

#
#  other SVGAs
#
#s3v864:
##   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x5333, Device_Id -
0x88c0, Driver_Name - svision
#
#ati32:
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x1002, Device_Id -
0x4158, Driver_Name - ati_vga, Type - C, Adpt_Config - N

#
# VBA_Option = Manufact_Name Product_Name Bus_Instance
Driver_Name
#             Driver_Instance Csr1 Csr2 Vector Bus_Prioity
#             Type Adpt_Config
#
vb:
#
# %%%VB
#
   VBA_Option = Manufact_Name - 'Digital', Product_Name - 'VME
Backplane Network Driver',  Bus_Instance - 0, Driver_Name - vb,
Driver_Instance - 0, Csr1 - 0, Csr2 - 0, Vector - 0x1150, Bus_Priority - 7,
Type - C, Adpt_Config - N
#
# pfm: Pseudo driver needed for performace measurements ... kprofile()
and/or uprofile()
#
pfm:
  Module_Config_Name = pfm
  Device_Dir = /dev
  Device_Char_Major = Any
  Device_Char_Minor = 0
  Device_Char_Files = pfcntr
  Device_User = root
  Device_Group = 0
  Device_Mode = 444
  Device_Major_Req = Same

#
# USB drivers.  Supported options are: Driver_Name, Vendor_Id,
Product_Id,
#             Release_Num, Device_Class, Device_Sub_Class,
Device_Protocol,
#             Configuration_Value, Interface_Number, Interface_Sub_Class,
#             Interface_Protocol, Interface_Class
DEC_USBhub:
                USB_Class_Driver = Driver_Name - usb_hub, Release_Num -
0x1000, Device_Class - 0x09, Device_Sub_Class - 0x1, Config_Device -
0x1
                USB_Class_Driver = Driver_Name - usb_hub, Release_Num -
0x1000, Device_Class - 0x09, Device_Sub_Class - 0x0, Config_Device -
0x1
DEC_USBmouse:
                USB_Class_Driver = Driver_Name - usb_mouse,
Release_Num - 0x1000, Device_Class - 0xff, Interface_Class - 0x03,
Interface_Sub_Class - 0x01, Interface_Protocol - 0x02, Config_Interface -
0x1
DEC_USBkeyboard:
                USB_Class_Driver = Driver_Name - usb_keyboard,
Release_Num - 0x1000, Device_Class - 0xff, Interface_Class - 0x03,
Interface_Sub_Class - 0x01, Interface_Protocol - 0x01, Config_Interface -
0x1
DEC_USBhid:
                USB_Class_Driver = Driver_Name - usb_hid, Release_Num -
0x1000, Device_Class - 0xff, Interface_Class - 0x03, Interface_Sub_Class -
0x01, Config_Interface - 0x1

#
# KGPSA Fibre Channel Adapter
#
emx:
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x10df, Device_Id -
0x1ae5, Driver_Name - emx, Type - A, Adpt_Config - N
#   PCI_Option =  PCI_SE_Rev - 0x210, Vendor_Id - 0x10df, Device_Id -
0xf700, Driver_Name - emx, Type - A, Adpt_Config - N

#
# Turning on lockmode=4, rt_preempt_opt=1, kmem_debug=0xe,
kmem_audit_count=5000
#
generic:
                lockmode = 2
                rt_preempt_opt = 0
                kmem_debug = 0
                kmem_audit_count = 1024
                memberid = 0
                new_vers_high = 1445681868665016768
                new_vers_low = 52314
                version_avendor = COMPAQ
                version_banner = Compaq Tru64 UNIX
                version_product = Tru64 UNIX
                version_vendor = Compaq Computer Corporation


ipc:
                msg_max = 32768
                msg_mnb = 30000
                msg_mni = 256
                msg_tql = 1024
                sem_aem = 32768
                sem_mni = 1024
                sem_msl = 2000
                sem_opm = 2000
                sem_ume = 2000
                sem_vmx = 320000
                shm_allocate_striped = 0
                shm_max = 63957085184
                shm_mni = 1024
                shm_seg = 512
                ssm_threshold = 0

rt:
                aio-max-retry = 2
                aio_task_max_num = 1024

proc:
                max-per-proc-address-space = 626279215104
                max_per_proc_data_size = 626279215104
                max_per_proc_stack_size = 13626279215104
                max_proc_per_user = 2048
                max_threads_per_user = 2048
#               max_users = 2048
                per_proc_address_space = 626279215104
                per_proc_data_size = 626279215104
                per_proc_stack_size = 158388608

net:
                netisrthreads = 2
#               netisrthreads = 5  (1 + # cpu's)

inet:

vm:
```

```
                dump_user_pte_pages = 1
                gh-chunks = 8063
                new_wire_method = 1
                swapdevice = /dev/disk/dsk2b
                vm-swap-eager = 0
#       vm_bigpg_enabled = 1
#       vm_bigpg_anon = 64
#       vm_bigpg_shm = 64
#       vm_bigpg_ssm = 64
#       vm_bigpg_stack = 64
#       vm_bigpg_seg = 64


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
```

ckpt.v1

```
Warmup Checkpoint
-------------------------

Start Warmup Checkpoint:
                Fri Jul 26 01:02:15 EDT 2002

Finished Warmup Checkpoint:
                Fri Jul 26 01:13:40 EDT 2002



Measurement Checkpoints
-------------------------

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

Start Checkpoint 1:
                Fri Jul 26 01:32:15 EDT 2002

Finished Checkpoint 1:
                Fri Jul 26 01:48:37 EDT 2002

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


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

Start Checkpoint 2:
                Fri Jul 26 02:02:15 EDT 2002

Finished Checkpoint 2:
                Fri Jul 26 02:18:55 EDT 2002

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


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

Start Checkpoint 3:
                Fri Jul 26 02:32:15 EDT 2002

Finished Checkpoint 3:
                Fri Jul 26 02:48:59 EDT 2002


************************************
************************************
Start Checkpoint 4:

                Fri Jul 26 03:02:15 EDT 2002

Finished Checkpoint 4:
                Fri Jul 26 03:18:54 EDT 2002

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

Appendix E

Auditor Attestation

Benchmark Sponsor:

Dave Stanley
Manager, Systems Quality &
Performance Engineering
Hewlett-Packard
110 Spit Brook Rd
Nashua, NH 03062-2698

Prasanta Gosh
Performance Manager
Sybase, Inc.
5000 Hacienda Drive
Dublin, CA 94568

July 31, 2002

I verified the TPC Benchmark™ C performance for the following Client/Server configuration:

Platform:	**HP AlphaServer ES45 Model 68/1250 4 CPU**
Operating system:	**HP Tru64 UNIX V5.1B**
Database Manager:	**Sybase Adaptive Server Enterprise 12.5**
Transaction Manager:	**BEA Tuxedo 6.5 CTS**

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	**tpmC**
Server: HP AlphaServer ES45 Model 68/1250 4 CPU				
4 x Alphachip EV68CB 21264C (1250 MHz)	32 GB Main (16MB Cache per processor)	280 x 9.1 GB 17 x 18 GB	1.33 Seconds	**56,375**
Eight (8) Compaq Proliant ML350 (Specification for each)				
2 x Pentium III (1 GHz)	640 MB Main (512 KB L2 Cache per processor)	1 x 18.2 GB	n/a	n/a

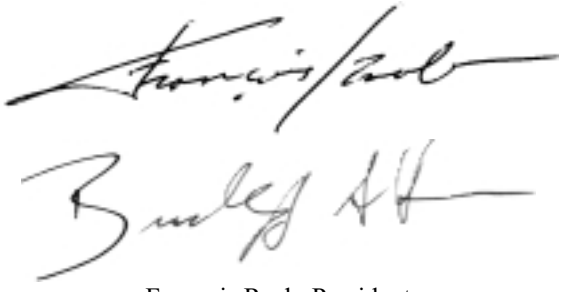
In my opinion, these performance results were produced in compliance with the TPC's requirements for the benchmark. The following verification items were given special attention:

- 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 (7200 seconds)
- The reported measurement interval was representative of steady state conditions
- Four checkpoints were taken during the reported measurement interval
- The 60 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

The measured system included eight Compaq Proliant 1600 front end systems that were substituted by eight Compaq Proliant ML350 systems in the priced configuration. Based on the specifications of these systems, it is my opinion that this substitution does not have a material effect on the reported performance.

Respectfully Yours,



François Raab, President

Bradley J. Askins, Auditor

Appendix F

Price Quotations

July 31, 2002 **Quote No: CHP072402**
Hewlett-Packard Company
Atten: Maria Lopez

Description	Part Number	Qty	Unit Price	Extension	*Service*
Server Hardware					
AS ES45 68/1250 M2 4GB Unix	DA-68EBA-DA	1	53456	53456	
ES45 Tower Enclosure	BA61M-CT	1	325	325	
ES45 68/1000 SMP CPU Unix	KN610-EB	3	11050	33150	
ES45 4GB Memory Option	MS620-DA	7	17306	121142	
PCI to GbE SX Adapter	DEGPA-SA	1	1267	1267	
Power Supply, Self-Sensing	H7906-A9	2	813	1626	
SmartAray 5304A/128 for Alpha	3X-KZPDC-DF	6	1624	9744	
3YR 24X7/4HR ES45 M2	*FM-4V724-36*	*1*	*17896*		*17,896*
VT510,White,North Amer, No Kedy	VT510-DA	1	397	397	
US/Canada Keyboard	PCXLA-NA	1	18	18	
17-03212-05 8MP-8MP Patch Cable	BN25G-07	1	6	6	
SubTotal Server:				**$221,131**	***$17,896***
STORAGE					
Ultra 68VHD 10M Cable Assembly	BN37A-03	24	91	2184	
Star Lite Storage Shelf	DS-SL13R-BA	24	2290	54960	
12/24GB 4mm Dat 5.25"	TLZ10-LB	1	444	444	
3YR 7X24/4HR,4MM DAT TAPE DRV	*FM-4M724-36*	*1*	*862*		*862*
9GB 10K U3 UNI HP HARD DRIVE**	3R-A0584-AA	308	207	63756	
18GB 10K U3 UNI HP HARD DRIVE **	3R-A0585-AA	19	207	3933	
Subtotal Storage:				**$125,277**	***$862***
Server Software					
3YR, AS ES40/45 UNIX BRNZ24X7	*FM-E4WUS-36*	*1*	1896		*1,896*
3YR AS ES45 UNIX SMP	*FM-62USM-36*	*3*	214		*642*
3yr Digital Unix O/S & LP	*FM-CDDST-36*	*1*	6765		*6,765*
Tru64 UNIX AlphaCDROM	QA-MT4AA-H8	1	293	293	
Subtotal Server Software:				**$293**	***$9,303***
Client Hardware					
Compaq ProLiant ML350 1.0Ghz	225561-001	8	1587	12696	
3YR 24X7 4HR WORKGROUP SVR	*FM-LO724-36*	*8*	*1160*		*9,280*
Pentium II P1000-256K Processor	207068-B21	8	571	4568	
512MB SDRAM DIMM Memory	128279-B21	8	435	3480	
ProLiant ML350 Hot Plug Drive Cage	161275-B21	8	413	3304	
18.2gb Wide Ultra 3 Drive	3R-A0585-AA	8	207	1656	
Compaq NC3134 Fast Ethernet NIC 64 Pci Dual Port 10/100	138603-B21	16	207	3312	
NC3134	138604-B21	8	207	1656	
Gigabit Daughter CD Upgrade	338456-B23	8	516	4128	
128MB SDRAM DIMM Memory Option	313615-B21	8	144	1152	
SC-SC Dual FO Cbl, MM, PP, 4.5M	BN34B-4E	8	62	496	
V700 15" Monitor NH US	261602-001	10	130	1300	
Subtotal Client Hardware:				**$37,748**	***$9,280***
Grand Total:				**$384,449**	***$37,341***

The Compaq AlphaServer ES45 carries a three (3) year, 7x24/4HR response warranty. The storage have a three (3) year on-site, 4-hour, 7-day per week response with a three (3) year return to manufacture warranty. All other products carry a standard warranty of three (3) years on-site; 4-hour x 7-days per week.

Traditional Compaq Intel Based Client Hardware Priced at US1plus%

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

THE ECOMMERCE TRANSACTION PLATFORM

July 30, 2002

Mr. Joseph Donaher
TPC-C Performance Project Manager
Hewlett-Packard Company
110 Spit Brook Road
Nashua, NH 03062-2698
603-884-2784

Dear Mr. Donaher:

Per your request I am enclosing the pricing information regarding TUXEDO 6.5 that you requested. This pricing applies to Tuxedo 6.4, 6.5, 7.1, and 8.0. Please note that Tuxedo 8.0 is our most recent version of Tuxedo. Core functionality services pricing is appropriate for your activities. As per the table below Compaq systems are classified as either a Tier 1, 2, 3, 4 or 5 systems depending on the performance and CPU capacity of the system. This quote is valid for 60 days from the date of this letter.

Tuxedo Core Functionality Services (CFS) Program Product Pricing and Description

TUX-CFS provides a basic level of middleware support for distributed computing, and is best used by organizations with substantial resources and knowledge for advanced distributed computing implementations.

TUX-CFS prices are server only and are based on the overall performance characteristics of the server and uses the same five tier computer classification as TUXEDO 6.4, 6.5, 7.1, and 8.0. Prices range from $3,000 for Tier 1 to $250,000 for Tier 5. Under this pricing option EVERY system running TUX-CFS at the user site must have a TUXEDO license installed and pay the appropriate per server license fees. Note that a 5% discount will apply to total list price license purchases less than $100,000 (for instance 30 Tier 1 servers – 30* 3,000 = $90,000 - would be eligible for a 5% discount). Support is not discountable.

Very Truly Yours,

Robert J. Gieringer

Rob Gieringer,
Worldwide Pricing Manager

BEA Tux/CFS Unlimited User License Fees Per Server

Unlimited User License fees per server	Number of Users	Dollar Amount	Maintenance (5 x 8) per year	Maintenance (7 x 24) per year
Tier 1 -- PC Servers with 1 or 2 CPUs, entry level RISC Uni-processor workstations and servers	Unlimited	$3,000.00	$480.00	$690.00
Tier 2 - PC Servers with 3 or 4 CPUs, Midrange RISC Uni-processor servers and workstations with up to 2 CPUs	Unlimited	$12,000.00	$1,920.00	$2,760.00
Tier 3 - Midrange Multiprocessors, up to 8 CPUs per system capacity	Unlimited	$30,000.00	$4,800.00	$6,900.00
Tier 4 - Large (more than 8, less than 32 CPUs)	Unlimited	$100,000.00	$16,000.00	$23,000.00
Tier 5 - Massively Parallel Systems, > 32 processors	Unlimited	$250,000.00	$40,000.00	$57,500.00

		Tier 1	Tier 1	Tier 2	Tier 3	Tier 4	Tier 5
Platform	**Operating System**						
Compaq Alpha	Open VMS Windows NT Digital UNIX		200 4/233 250 4/266 300 all 400 4/166 800 all 1000 4/233, XP900, au500, au600	600 5/300 2000, 2100 4/233, 1000A 1200, 3000 Server 53XX, DS10,DS10L, DS20,DS20E XP1000	2100 4/275 2100 5/250 4000 5/300 4000 5/400 4000 5/466 7300/7300R, 7305/7305R 8200 5/300 8200 5/300 4100 5/300 4100 5/400 ES40,ES45	Alpha 8400 5/330, 5/440, GS60, GS60E, GS80	Alpha Server 8400 5/625 or later GS140, GS160, GS320
Compaq Proliant	Windows NT	ML330,ML350, ML370,ML530,4 00,800, NeoServer	1600, 1850, 3000, DL320,DL360,D L380	5500, 6000, 6400, 6500, L570,DL580, DL590,CL380	DL760,7000, 8000, 8500, ML750		

ENTERASYS NETWORKS

Quotation

Date	7/29/02			**Sales Rep**	Allison R Strome		
Quote #	1-R48RS	**Rev #**	1	**Phone**	(603) 337-2125	**Fax**	(603) 337-2611
Valid From:	07/29/2002	**To**	08/28/2002	**Contract**		**SAP**	
Type	Hardware/Software/Service	**Duration**		**Contract Term**		**to**	
PO#		**Schedule**		**OSR**	RHOLLAND		

To Maria Lopez
(603) 884-0030 Fax () -
Hewlett-Packard
110 Spit Brook Road
Nashua, NH 03062

Line	Product Description	Qty.	Enterasys List	Net Price	Extended Price
1	VH-8G 8 port Gigabit 1000Base-SX Std alone fixed SC connectors	1	$4,995.00	$2,747.25	$2,747.25

Total - Products	**$2,747.25**
Total - Maintenance	**$0.00**
Total - ProServ	**$0.00**
Grand Total	**$2,747.25**

ALL SALES ARE SUBJECT TO ENTERASYS'S STANDARD TERMS AND CONDITIONS FOR PURCHASE AND SUPPORT SERVICE. A COPY OF THE STANDARD TERMS IS AVAILABLE UPON REQUEST.

In order to process your order as quickly as possible please include the following on all purchase orders: Quote number, Requested delivery date, Freight carrier and Billing & Shipping addresses with contact name and phone number.

1-R48RS

1