Hewlett-Packard Company

TPC Benchmark™ C
Full Disclosure Report
for
HP ProLiant DL380 G7
using
Microsoft SQL Server 2005 Enterprise x64 Edition SP3
and
Windows Server 2008 R2 Enterprise Edition

First Edition
Submitted for Review
May 4, 2011

First Edition –May 2011

Hewlett-Packard Company (HP) 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. HP 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, HP provides no warranty of the pricing information in this document.

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

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

Printed in U.S.A., 2011
HP ProLiant DL380 G7 and ProLiant are registered trademarks of Hewlett-Packard Company.

Microsoft, Windows Server 2003, Windows Server 2008 R2 and SQL Server 2005 x64 are registered trademarks of Microsoft Corporation.

Xeon is a registered trademark of Intel.

TPC Benchmark is a trademark of the Transaction Processing Performance Council.

Other product names mentioned in this document may be trademarks and/or registered trademarks of their respective companies.

Table of Contents

Preface

The TPC Benchmark C was developed by the Transaction Processing Performance Council (TPC). The TPC was founded to define transaction processing benchmarks and to disseminate objective, verifiable performance data to the industry. This full disclosure report is based on the TPC Benchmark C Standard Specifications Version 5.11

TPC Benchmark C Overview

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

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

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

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

Although these specifications express implementation in terms of a relational data model with conventional locking scheme, the database may be implemented using any commercially available database management system (DBMS), database server, file system, or other data repository that provides a functionally equivalent implementation. The terms "table", "row", and "column" are used in this document only as examples of logical data structures.

TPC-C uses terminology and metrics that are similar to other benchmarks, originated by the TPC or others. Such similarity in terminology does not in any way imply that TPC-C results are comparable to other benchmarks. The only benchmark results comparable to TPC-C are other TPC-C results conformant with the same revision.

Despite the fact that this benchmark offers a rich environment that emulates many OLTP applications, this benchmark does not reflect the entire range of OLTP requirements. In addition, the extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

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

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark C test conducted on the HP ProLiant DL380 G7. The operating system used for the benchmark was Windows Server 2008R2 Enterprise Edition. The DBMS used was Microsoft SQL Server 2005 Enterprise x64 Edition SP3.

TPC Benchmark C Metrics

The standard TPC Benchmark C metrics, tpmC (transactions per minute), and price per tpmC (three year capital cost per measured tpmC), and the availability date are reported as:

1,024,380 tpmC
USD $0.65 per tpmC

The availability date is June 20, 2011.

Standard and Executive Summary Statements

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

Auditor

The benchmark configuration, environment and methodology were audited by Lorna Livingtree of Performance Metrics, Inc. to verify compliance with the relevant TPC specifications.

General Items

Test Sponsor

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

This benchmark was sponsored by Hewlett-Packard Company. The benchmark was developed and engineered by Hewlett-Packard Company. Testing took place at HP benchmarking laboratories in Houston, Texas.

Application Code and Definition Statements

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 output functions.

Appendix A contains all source code implemented in this benchmark.

Parameter Settings

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

- *Database options*
- *Recover/commit options*
- *Consistency locking options*
- *Operating system and application configuration parameters*

This requirement can be satisfied by providing a full list of all parameters.

Appendix C contains the tunable parameters to for the database, the operating system, and the transaction monitor.

Configuration Items

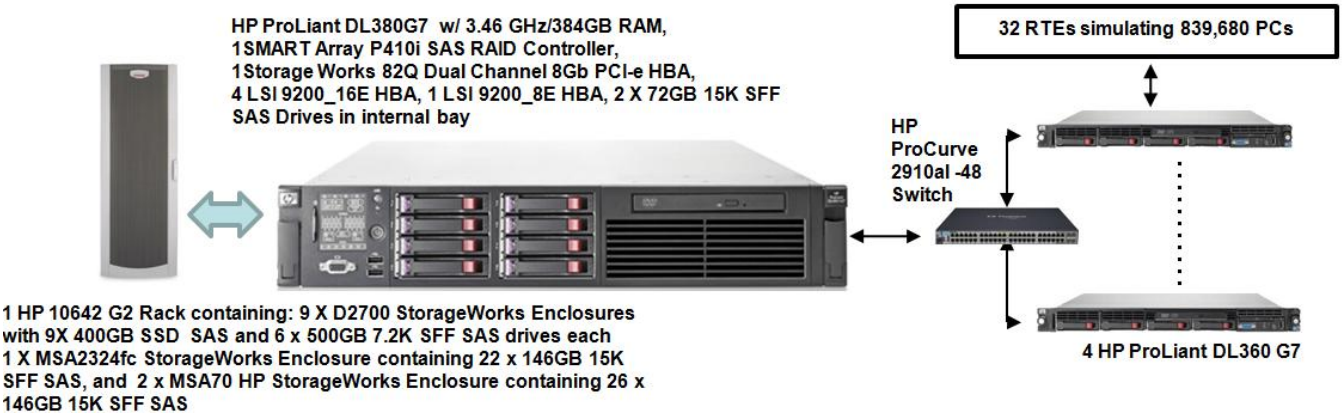
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences.

The configuration diagram for both the tested and priced systems are included on the following page.

Hewlett-Packard Company	HP ProLiant DL380 G7 3.46 GHz 12 MB Cache C/S with 4 HP ProLiant DL360 G7	TPC-C Rev. 5.11 TPC-Pricing 1.6.0 **Report Date: May 4, 2011**

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	TPC-Energy Metric
USD $661,902	**1,024,380 tpmC**	**USD $0.65**	**June 20, 2011**	**N/R**

Database Server Processors /Cores/Threads	Database Manager	Operating System	Other Software	Number of Users
2/12/24 Intel Xeon X5690 3.46 GHz 12MB cache	Microsoft SQL Server 2005 Enterprise x64 Edition SP3	Windows Server 2008 R2 Enterprise Edition	Microsoft Visual Studio 2008 Microsoft COM+	**839,680**



	Server		Each Client	
System Components	Quantity	Description	Quantity	Description
Processors/Cores/Threads	2	Intel Xeon X5690 3.46GHz 12MB cache - 6 cores /12 threads	2/6/12	2.93 GHz Intel Xeon X5670 w/ 12MB cache
Memory	384 GB	12 x 16GB DDR3 6 x 32GB DDR3	24GB	12 x 2048 MB
Disk Controllers	1 4 1 1	Smart P410i Controller LSI 9200_16E HBA LSI 9200_8E HBA 82Q 8Gb Dual Channel	1	Integrated Smart Array P410i Controller
Disk Drives	81 54 48 2	400 GB SSD SAS 500 GB 6G SAS 7.2K SFF 146 GB 15K SFF SAS 72 GB 15K SFF SAS	4	300GB 10K SFF SAS
Total Storage		66,552 GB		555.36 GB

Hewlett-Packard Company	HP ProLiant DL380 G7			TPC-C Rev. 5.11		
				Report Date	4-May-11	
Description	Part Number	Brand	Unit Price	Qty	Extended Price	3 yr. Maint. Price
Server Hardware						
HP DL380G7 SFF CTO Chassis	583914-B21	1	1,622	1	1,622	
HP DL380 G7 Intel Xeon X5690 (3.46GHz) Processor Kit	587498-B21	1	2,299	2	4,598	
HP 16GB 2Rx4 PC3L-10600R-9 Kit	627812-B21	1	1,349	12	16,188	
HP 32GB 4Rx4 PC3L-8500R-7 Kit	627814-B21	1	5,999	6	35,994	
HP 72GB 6G SAS 15K SFF DP ENT HDD	512545-B21	1	279	2	558	
HP StorageWorks 82Q 8Gb Dual Port PCI-e Fibre Channel Host Bus Adapter	AJ764A	1	1,999	1	1,999	
HP Compaq LE1711 17-inch LCD Monitor	EM886AA#ABA	1	159	1	159	
HP PS/2 Keyboard And Mouse Bundle	RC464AA#ABA	1	39	1	39	
HP V142 600mm Pallet 100 series Rack	358254-B21	1	789	1	789	
HP 500GB 6G SAS 7.2K SFF DP ENT HDD	507610-B21	1	349	54	18,846	
HP 500GB 6G SAS 7.2K SFF DP ENT HDD (10% Spares)	507610-B21	1	349	6	2,094	
HP 400GB SSD SAS SFF	632430-002	1*	5,999	81	485,919	
HP 146GB 6G SAS 15K SFF DP ENT HDD	512547-B21	1	369	48	17,712	
HP StorageWorks P2000 G3 MSA FC Dual Controller SFF Modular Smart Array	AP846A	1	10,350	1	10,350	
HP StorageWorks D2700 Disk Enclosure (10% Spares)	AJ941A	1	3,399	1	3,399	
HP StorageWorks MSA70 Array	418800-B21	1	3,199	2	6,398	
HP 3y 4h 24x7 D2000 Encl HW Support,MSA 60/70 Enclosures	UF303E	1	1,982	2		3,964
HP 3y 4h 24x7 D2000 Encl HW Support,D2000 Enclosures	UQ540E	1	1,980	9		17,820
HP 3y 4h 24x7 MSA 2000 G3 HWSupp,MSA 2000 G3 Array	UV394E	1	2,079	1		2,079
4-Hour On-site Service, 7-Day x 24-Hour Coverage, 3 Years , DL380	U4608E	1	1,309	1		1,309
LSI 9200_16e	LSI00189	4	528	4	2,112	
LSI 9200_16e (10% spares)	LSI00189	4	528	2		1,056
LSI 9200_8e	LSI00188	4	355	1	355	
LSI 9200_8e (10% spares)	LSI00188	4	355	2		710
				Subtotal	609,131	26,938
Server Software						
Microsoft SQL Server 2005 Enterprise X64 Edition(per processor)	810-03134	2	23,432	2	46,864	Incl Below
Visual Studio 2008 Standard Edition	127-00166	2	275	1	275	Incl Below
Microsoft Windows Server 2008 R2 Enterprise Edition	P72-04217	2	2,280	8	18,240	Incl Below
Microsoft Problem Resolution Services		2	259	1		259
				Subtotal	65,379	259
Client Hardware						
HP DL360G7 CTO Chassis	579237-B21	1	1,471	4	5,884	
HP X5670 DL360G6/G7 FIO Kit	588062-L21	1	1,999	8	15,992	
HP 460W CS HE Power Supply Kit	503296-B21	1	229	4	916	
HP NC365T 4-port Ethernet Server Adapter	593722-B21	1	439	8	3,512	
HP 2GB 2Rx8 PC3-10600R-9 Kit	500656-B21	1	110	48	5,280	
HP 300GB 6G SAS 10K SFF (2.5-inch) Dual Port Enterprise 3yr Warranty	507127-B21	1	409	16	6,544	
HP Compaq LE1711 17-inch LCD Monitor	EM886AA#ABA	1	159	4	636	
HP PS/2 Keyboard And Mouse Bundle	RC464AA#ABA	1	39	4	156	
HP CP 3Y 4H 24x7 HW Entry300 4-Hour 24 Hour x 7 Day Coverage 3 Years	U4497E	1	750	4		3,000
				Subtotal	38,920	3,000
Client Software						
Windows Server 2008 R2 Enterprise Edition	P72-04217	2	2,280	8	18,240	Incl. Above
Windows Server 2008 R2 Standard Edition	P73-04980	2	711	4	2,844	Incl. Above
				Subtotal	21,084	0
User Connectivity						
HP ProCurve 2910al-48G Switch	J9147A#ABA	1	4,569	1	4,569	
HP ProCurve3 Yr 4 hr/24x7 Onsite	H2893E	1	1,307	1		1,307
CAT 6 7 Foot White Patch Cable	CB242-7W	3	2	32	50	
CAT 6 7 Foot White Patch Cable (spares)	CB242-7W	3	2	4		6
				Subtotal	4,619	1,313
Large Purchase and Net 30 discount (See Note 1)	16.0%	1			($104,024)	($4,717)
				Total	$635,108	$26,794

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 pricing 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.

Three-Year Cost of Ownership: USD $661,902

tpmC Rating: **1,024,380**

$ / tpmC: USD **$0.65**

Pricing: 1=HP Direct 800-203-6748 2= Microsoft 3= deepsurplus.com 4= Microland Electronics

Note 1 = Discount based on HP Direct guidence applies to all lines where pricing = 1

Note 2 = * These components are not immediately orderable. For price verification before order date: e-mail hp.pricing.desk@hp.com

Note 3 = The benchmark results were audited by Lorna Livingtree of Performance Metrics

Numerical Quantities Summary

MQTH, Computed Maximum Qualified Throughput 1,024,380 tpmC

Response Times (in seconds)	Average	90%	Maximum
New-Order	0.80	2.60	85.58
Payment	0.80	2.59	85.72
Order-Status	0.79	2.58	60.63
Delivery (interactive portion)	0.22	0.48	74.18
Delivery (deferred portion)	0.05	0.08	4.94
Stock-Level	0.80	2.60	9.01
Menu	0.22	0.48	83.89

Transaction Mix, in percent of total transaction	
New-Order	44.95%
Payment	43.02%
Order-Status	4.02%
Delivery	4.00%
Stock-Level	4.01%

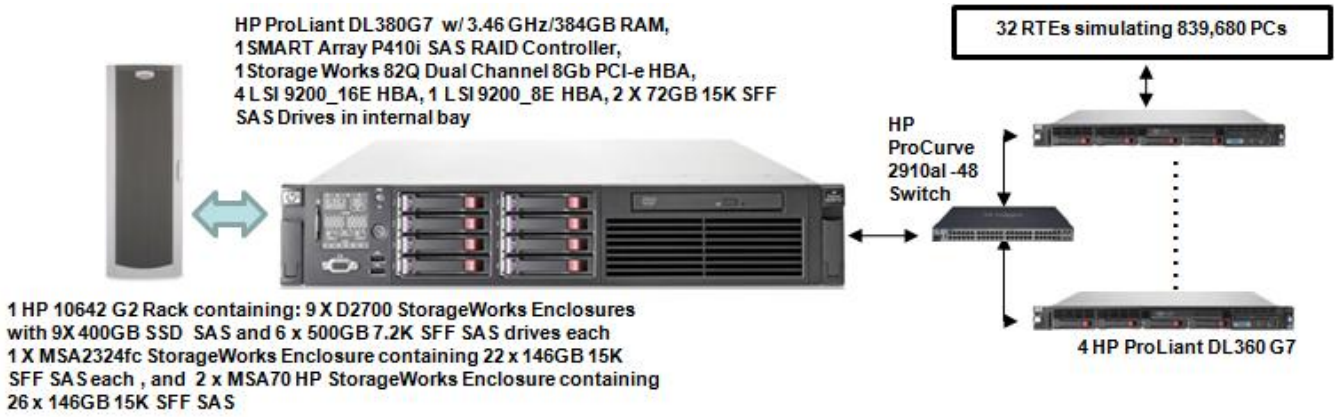
Emulation Delay (in seconds)	Resp.Time	Menu
New-Order	0.10	0.10
Payment	0.10	0.10
Order-Status	0.10	0.10
Delivery (interactive)	0.10	0.10
Stock-Level	0.10	0.10

Keying/Think Times (in seconds)	Min.	Average	Max.
New-Order	18.02/0.00	18.03/12.11	18.05/120.92
Payment	3.02/0.00	3.03/12.10	3.05/120.93
Order-Status	2.02/0.00	2.03/10.06	2.05/100.83
Delivery (interactive)	2.02/0.00	2.03/5.09	2.05/50.72
Stock-Level	2.02/0.00	2.03/5.07	2.05/50.72

Test Duration	
Ramp-up time	62 minutes
Measurement interval	120 minutes
Transactions (all types) completed during measurement interval	273,475,276
Ramp down time	2 minutes

Checkpointing	
Number of checkpoints	11
Checkpoint interval	30 minutes

Figure 1. Benchmarked and Priced Configuration



Clause 1 Related Items

Table Definitions

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

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

Physical Organization of Database

The physical organization of tables and indices within the database must be disclosed.
The tested configuration consisted of 91 SSD drives at 400GB for database data, two 72GB drives for the operating system, 48 drives at 146GB for database log and 54 drives at 500 GB for backup and 60 day space. There were 4 x LSI 9200-16e and 1 x LSI 9200-8e controllers connected to 9 x D2700 storage boxes with 9 x 400GB SSD drives each for database data and 6 x 500GB drives each for backup, 48 x 146GB drives on one MSA2324fc controller connected to 2 MSA70 storage box for database log, and 2 x 72GB drives on the SMART P410i controller for the operating system.

Benchmarked Configuration:

SMART-P400i Controller, Slot 0, Array A, disk 0
LOGICAL DRIVE C: Total Capacity = 68.23 GB RAID 1
Microsoft Windows Server 2008 R2 Enterprise Edition

HP 82Q, Slot 6 connected to MSA2324fc, disk 1-4

Disk 1: LOGICAL DRIVE E:	Total Capacity = 1092.44 GB	RAID 1+0
Disk 2: LOGICAL DRIVE F:	Total Capacity = 1092.44 GB	RAID 1+0
Disk 3: LOGICAL DRIVE G:	Total Capacity = 1092.44 GB	RAID 1+0
Disk 4: LOGICAL DRIVE H:	Total Capacity = 1092.44 GB	RAID 1+0

Logical drives E, F, G, and H are software stripe volumes created over 2 physical RAID 1+0 volumes each.

LSI 9200_16E, Slot 1, disk 5-13
LOGICAL DRIVE C:\dev\tpcc_1-18: Total Capacity = 1.276 TB RAID 0

LSI 9200_16E, Slot 1, disk 14-19
LOGICAL DRIVE C:\backup\1: Total Capacity = 2.794 TB RAID 0

LSI 9200_16E, Slot 1, disk 20-28
LOGICAL DRIVE C:\dev\tpcc_19-36: Total Capacity = 1.276 TB RAID 0

LSI 9200_16E, Slot 1, disk 29-34
LOGICAL DRIVE C:\backup\2: Total Capacity = 2.794 TB RAID 0

LSI 9200_16E, Slot 2, disk 35-43
LOGICAL DRIVE C:\dev\tpcc_37-54: Total Capacity = 1.276 TB RAID 0

LSI 9200_16E, Slot 2, disk 44-49
LOGICAL DRIVE C:\backup\3: Total Capacity = 2.794 TB RAID 0

LSI 9200_16E, Slot 2, disk 50-58
LOGICAL DRIVE C:\dev\tpcc_55-72: Total Capacity = 1.276 TB RAID 0

LSI 9200_16E, Slot 2, disk 59-64

LOGICAL DRIVE C:\backup\4:	Total Capacity = 2.794 TB	RAID 0

LSI 9200_16E, Slot 3, disk 65-73

LOGICAL DRIVE C:\dev\tpcc_73-90:	Total Capacity = 1.276 TB	RAID 0

LSI 9200_16E, Slot 3, disk 74-79

LOGICAL DRIVE C:\backup\5:	Total Capacity = 2.794 TB	RAID 0

LSI 9200_16E, Slot 3, disk 80-88

LOGICAL DRIVE C:\dev\tpcc_91-108:	Total Capacity = 1.276 TB	RAID 0

LSI 9200_16E, Slot 3, disk 89-94

LOGICAL DRIVE C:\backup\6:	Total Capacity = 2.794 TB	RAID 0

LSI 9200_16E, Slot 4, disk 95-103

LOGICAL DRIVE C:\dev\tpcc_109-126:	Total Capacity = 1.276 TB	RAID 0

LSI 9200_16E, Slot 4, disk 104-109

LOGICAL DRIVE C:\backup\7:	Total Capacity = 2.794 TB	RAID 0

LSI 9200_16E, Slot 4, disk 110-118

LOGICAL DRIVE C:\dev\tpcc_127-144:	Total Capacity = 1.276 TB	RAID 0

LSI 9200_16E, Slot 4, disk 119-124

LOGICAL DRIVE C:\backup\8:	Total Capacity = 2.794 TB	RAID 0

LSI 9200_8E, Slot 5, disk 125-133

LOGICAL DRIVE C:\dev\tpcc_145-162:	Total Capacity = 1.276 TB	RAID 0

LSI 9200_8E, Slot 5, disk 134-139

LOGICAL DRIVE C:\backup\9:	Total Capacity = 2.794 TB	RAID 0

Priced Configuration vs. Measured Configuration:

Priced configuration was identical to the measured configuration

Insert and Delete Operations

It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any restrictions in the SUT database implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the minimum key value for these new rows.

All insert and delete functions were fully operational during the entire benchmark.

Partitioning

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

No partitioning was used in this benchmark.

Replication, Duplication or Additions

Replication of tables, if used, must be disclosed. Additional and/or duplicated attributes in any table must be disclosed along with a statement on the impact on performance.

No replications, duplications or additional attributes were used in this benchmark.

Clause 2 Related Items

Random Number Generation

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

In the Benchcraft RTE from Microsoft, each driver engine uses an independent random number sequence. All of the users within a given driver draw from the same sequence.

The Benchcraft RTE computes random integers as described in “Random Numbers Generators: Good Ones Are Hard to Find." Communications of the ACM - October 1988 Volume 31 Number 10.

The seeds for each user were captured and verified by the auditor to be unique. In addition, the contents of the database were systematically searched, and randomly sampled by the auditor for patterns that would indicate the random number generator had affected any kind of a discernible pattern; none was found.

Input/Output Screen Layout

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

All screen layouts followed the specifications exactly.

Priced Terminal Feature Verification

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

The terminal attributes were verified by the auditor. The auditor manually exercised each specification on a representative HP ProLiant web server.

Presentation Manager or Intelligent Terminal

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

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

Transaction Statistics

Table 2.1 lists the numerical quantities that Clauses 8.1.3.5 to 8.1.3.11 require.

Table 2.1 Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	99.00%
	Remote warehouse order lines	1.00%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home warehouse payments	85.00%
	Remote warehouse payments	15.00%
	Accessed by last name	59.96%
Order Status	Accessed by last name	59.95%
Transaction Mix	New Order	44.95%
	Payment	43.02%
	Order status	4.02%
	Delivery	4.00%
	Stock level	4.01%

Queuing Mechanism

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

Microsoft COM+ on each client machine served as the queuing mechanism to the database. Each delivery request was submitted to Microsoft COM+ asynchronously with control being returned to the client process immediately and the deferred delivery part completing asynchronously.

The source code is listed in Appendix A.

Clause 3 Related Items

Transaction System Properties (ACID)

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

All ACID property tests were successful. The executions are described below.

Atomicity

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

Completed Transactions

A row was selected in a script from the warehouse, district and customer tables, and the balances noted. A payment transaction was started with the same warehouse, district and customer identifiers and a known amount. The payment transaction was committed and the rows were verified to contain correctly updated balances.

Aborted Transactions

A row was selected in a script from the warehouse, district and customer tables, and the balances noted. A payment transaction was started with the same warehouse, district and customer identifiers and a known amount. The payment transaction was rolled back and the rows were verified to contain the original balances.

Consistency

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

Consistency conditions one through four were tested using a script to issue queries to the database. The results of the queries verified that the database was consistent for all four tests.

A run was executed under full load lasting over two hours and included a checkpoint.

The script was executed again. The result of the same queries verified that the database remained consistent after the run.

Isolation

Sufficient conditions must be enabled at either the system or application level to ensure the required isolation defined above (clause 3.4.1) is obtained.

Isolation tests one through nine were executed using shell scripts to issue queries to the database. Each script included timestamps to demonstrate the concurrency of operations. The results of the queries were captured to files. The captured files were verified by the auditor to demonstrate the required isolation had been met.

In addition, the phantom tests and the stock level tests were executed and verified.

For Isolation test seven, case A was followed.

Durability

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

Durable Media Failure

Loss of Data and Log

To demonstrate recovery from a permanent failure of durable medium containing DBMS logs and TPC-C tables, the following steps were executed. This test was executed on a fully scaled database of 84,000 warehouses of which 83,968 were used under a load of 839,680 users.

- The total number of New Orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving the beginning count.
- The RTEs were started with 839,680 users.
- The test was allowed to run for a minimum of 5 minutes.
- One disk was removed from the P2000 Storage containing the log disks.
- Since the disk was mirrored, processing was not interrupted. This was verified by checking the user's status on the RTE.
- One of the data disks was removed from one D2700 data drive cabinet.
- When Microsoft SQL Server recorded errors about not being able to access the database, the RTE was shut down, and a database transaction log dump was taken.
- Microsoft SQL Server was shutdown, and the system rebooted after replacing the pulled drives with new drives.
- After the drive configuration was reset for data drive and RAID recovery for log drive Microsoft SQL Server was started.
- The database was restored from backup and the transaction log dump was applied.
- Consistency condition #3 was executed and verified.
- Step 2 was repeated and the difference between the first and second counts was noted.
- An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
- The counts in steps 12 and 13 were compared and the results verified that all committed transactions had been successfully recovered.
- Samples were taken from the RTE files and used to query the database to demonstrate successful transactions had corresponding rows in the ORDER table.

Instantaneous Interruption and Loss of Memory

Because loss of power erases the contents of memory, the instantaneous interruption and the loss of memory tests were combined into a single test. This test was executed on a fully scaled database of 84,000 warehouses, of which 83,968 warehouses were used, under a full load of 839,680 users. The following steps were executed:

- The total number of New Orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving the beginning count.
- The RTE was started with 839,680 users.
- The test was allowed to run for a minimum of 5 minutes.
- Pulling the power cords from the SUT induced system crash and loss of memory. No battery backup or Uninterruptible Power Supply (UPS) were used to preserve the contents of memory.
- The RTE was paused then stopped.
- Power was restored and the system restarted.
- Microsoft SQL Server was restarted and performed an automatic recovery.

- Consistency condition #3 was executed and verified.
- Step 1 was repeated and the difference between the first and second counts was noted.
- An RTE report was generated for the entire run time giving the number of NEW-ORDERS successfully returned to the RTE.
- The counts in step 9 and 10 were compared and the results verified that all committed transactions had been successfully recovered.
- Samples were taken from the RTE files and used to query the database to demonstrate successful transactions had corresponding rows in the ORDER table.

Clause 4 Related Items

Initial Cardinality of Tables

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

Table 4.1 Number of Rows for Server

Table	Cardinality as built
Warehouse	83,968
District	839,680
Customer	2,520,000,000
History	2,520,000,000
Orders	2,520,000,000
New Order	756,000,000
Order Line	25,199,922,151
Stock	8,400,000,000
Item	100,000
Unused Warehouses	32

Database Layout

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

The database data was housed on 81 SSD drives at 400GB, two 72GB drives for the operating system, 48 drives at 146GB for database log and 54 drives at 500 GB for backup and 60 day space. There were 4 x LSI 9200-16e controllers and 1 x LSI 9200-8e connected to 9 x D2700 storage boxes with 9 x 400GB SSD drives each for database data and 6 x 500GB drives each for backup, 48 x 146GB drives on one MSA P2000 G3 controller connected to 2 x MSA70 storage box for database log, and 2 x 72GB on the SMART P410i controller for the operating system. Section 1.2 of this report details the distribution of database tables across all disks. The code that creates the file groups and tables is included in Appendix B.

Type of Database

A statement must be provided that describes:

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

Microsoft SQL Server 2005 Enterprise x64 Edition is a relational DBMS.

The interface used was Microsoft SQL Server stored procedures accessed with Remote Procedure Calls embedded in C code.

Database Mapping

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

The database was not replicated.

60 Day Space

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

To calculate the space required to sustain the database log for 8 hours of growth at steady state, the following steps were followed:

- The free space on the log file was queried using *dbcc sqlperf(logspace)*.
- Transactions were run against the database with a full load of users.
- The free space was again queried using *dbcc sqlperf(logspace)*.
- The space used was calculated as the difference between the first and second query.
- The number of NEW-ORDERS was verified from the difference in the sum(d_next_o_id) taken from before and after the run.
- The space used was divided by the number of NEW-ORDERS giving a space used per NEW-ORDER transaction.
- The space used per transaction was multiplied by the measured tpmC rate times 480 minutes.

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

Details of both the 8-hour transaction log space requirements and the 60-day space requirements are shown in Appendix D.

Clause 5 Related Items

Throughput

Measured tpmC must be reported

Measured tpmC 1,024,380 tpmC
Price per tpmC USD $0.65

Response Times

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

Table 5.2: Response Times

Type	Average	90th %	Maximum
New-Order	0.80	2.60	85.58
Payment	0.80	2.59	85.72
Order-Status	0.79	2.58	60.63
Interactive Delivery	0.22	0.48	74.18
Deferred Delivery	0.05	0.08	4.94
Stock-Level	0.80	2.60	77.09
Menu	0.22	0.48	83.89

Keying and Think Times

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

Table 5.3: Keying Times

Type	Minimum	Average	Maximum
New-Order	18.02	18.03	18.05
Payment	3.02	3.03	3.05
Order-Status	2.02	2.03	2. 05
Interactive Delivery	2.02	2.03	2. 05
Stock-Level	2.02	2.03	2. 05

Table 5.4: Think Times

Type	Minimum	Average	Maximum
New-Order	0.00	12.11	120.92
Payment	0.00	12.10	120.93
Order-Status	0.00	10.06	100.83
Interactive Delivery	0.00	5.08	50.72
Stock-Level	0.00	5.09	50.72

Response Time Frequency Distribution Curves and Other Graphs

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

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

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

Keying Time frequency distribution curves (see Clause 5.6.4) must be reported for each transaction type.

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

Figure 3. New Order Response Time Distribution

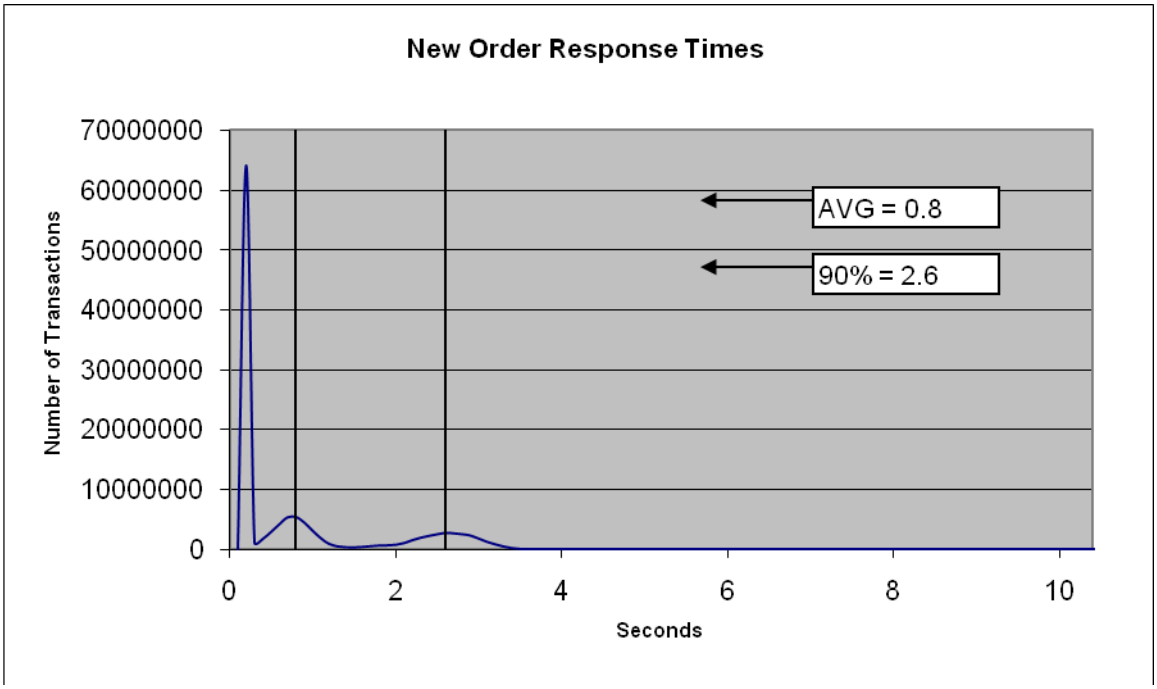


Figure 4. Payment Response Time Distribution

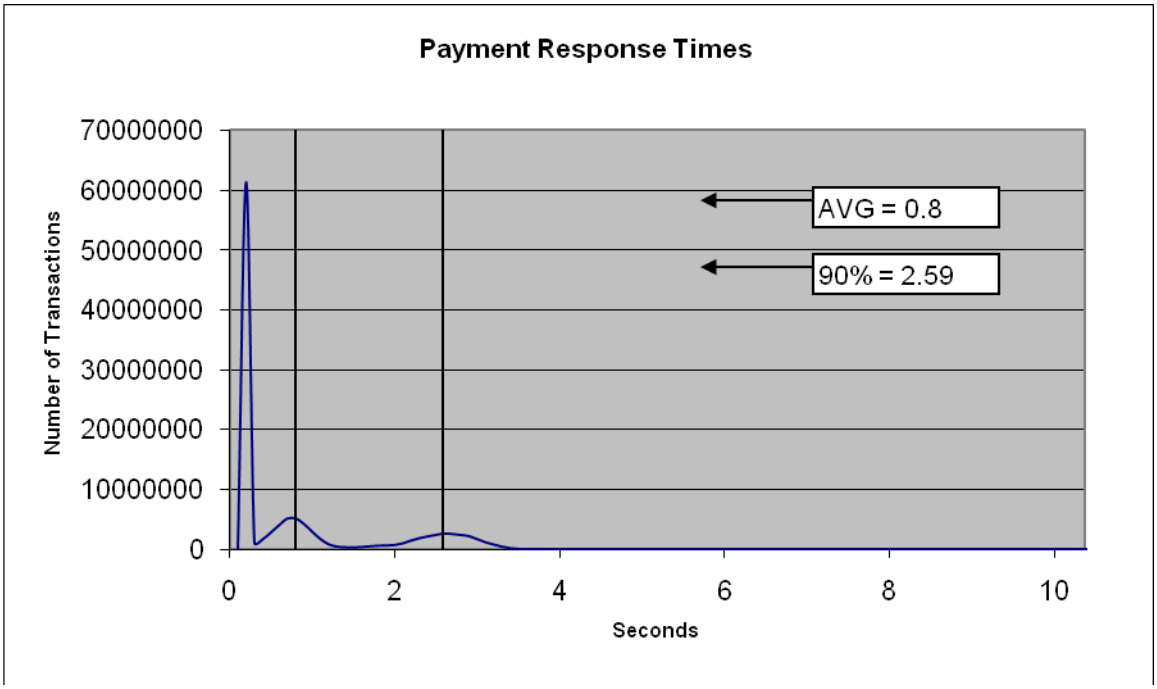


Figure 5. Order Status Response Time Distribution

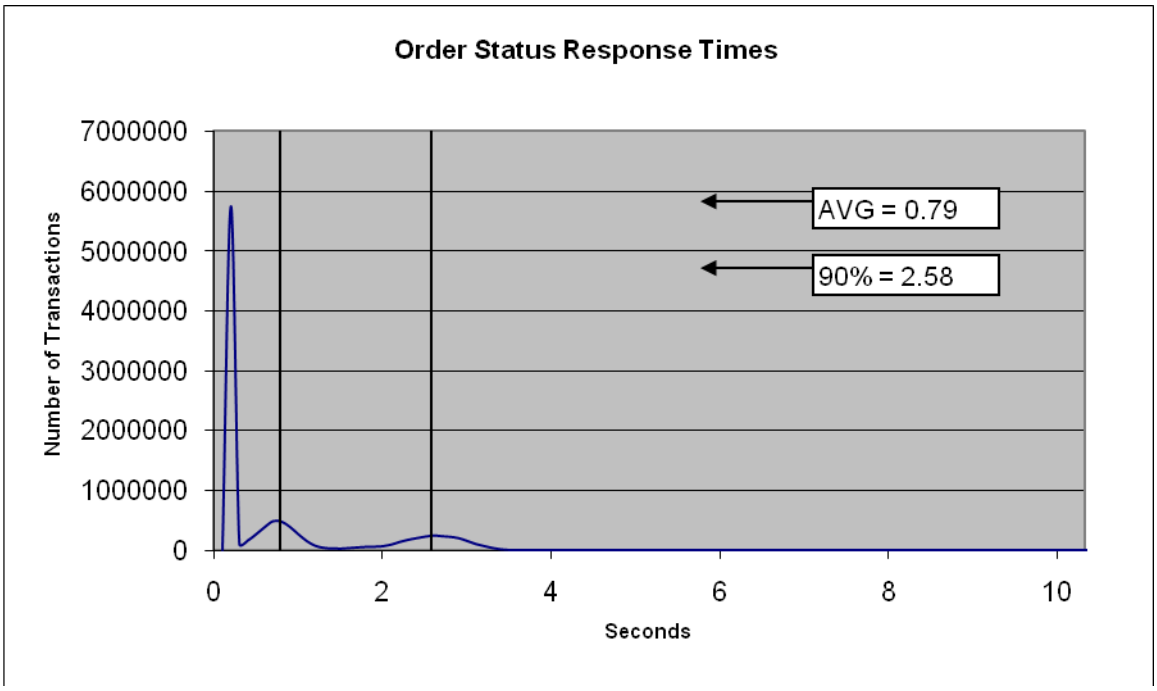


Figure 6. Delivery Response Time Distribution

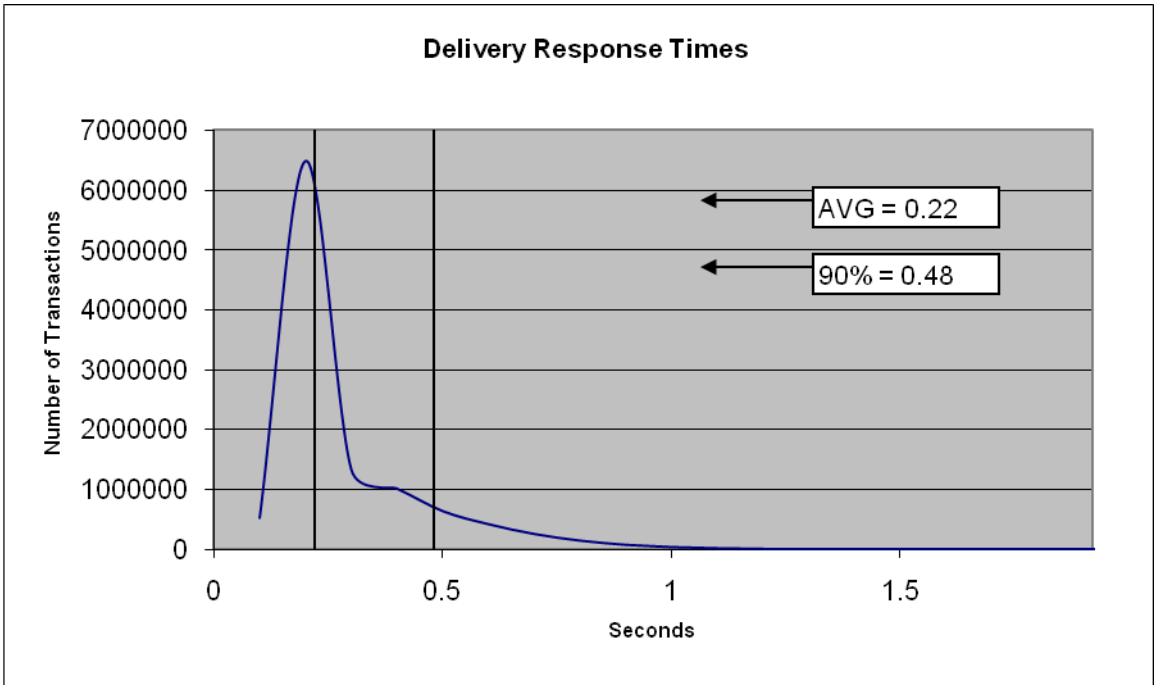


Figure 7. Stock Level Response Time Distribution

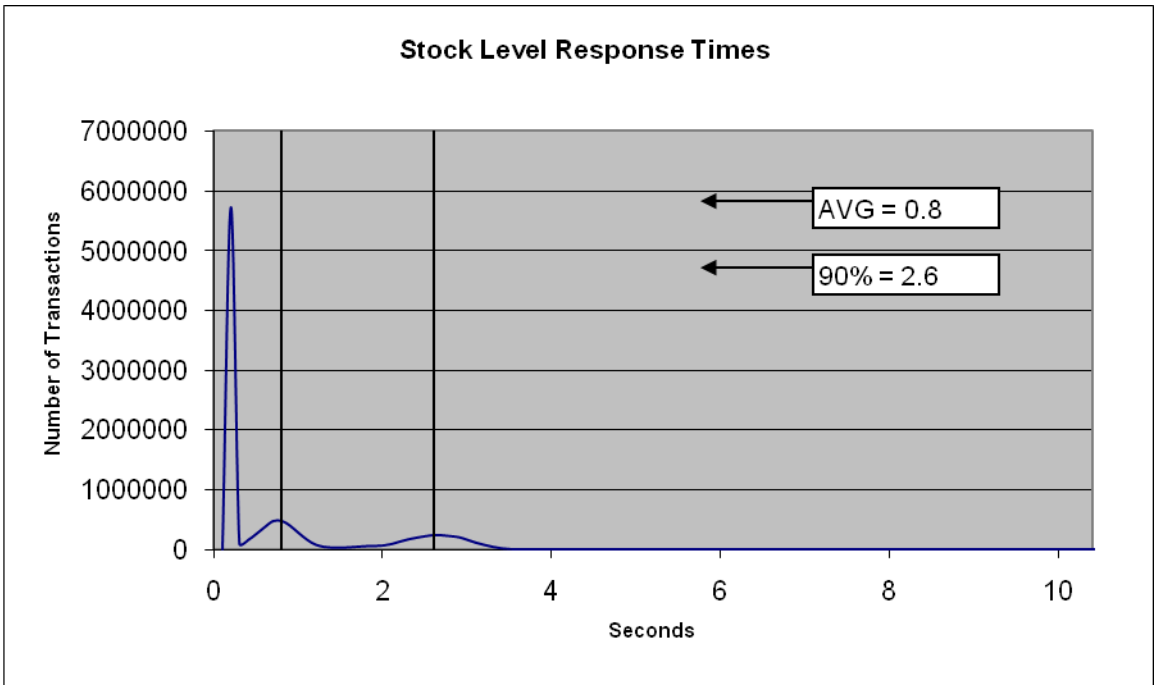


Figure 8. Response Time vs. Throughput

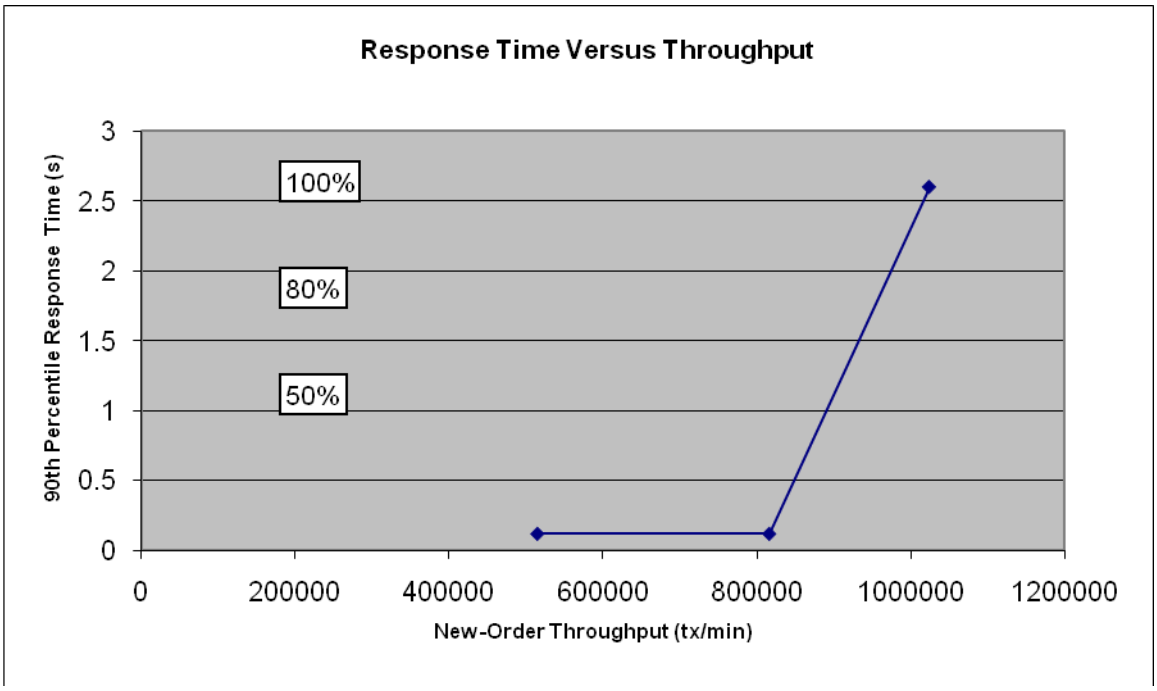


Figure 9. New Order Think Time Distribution

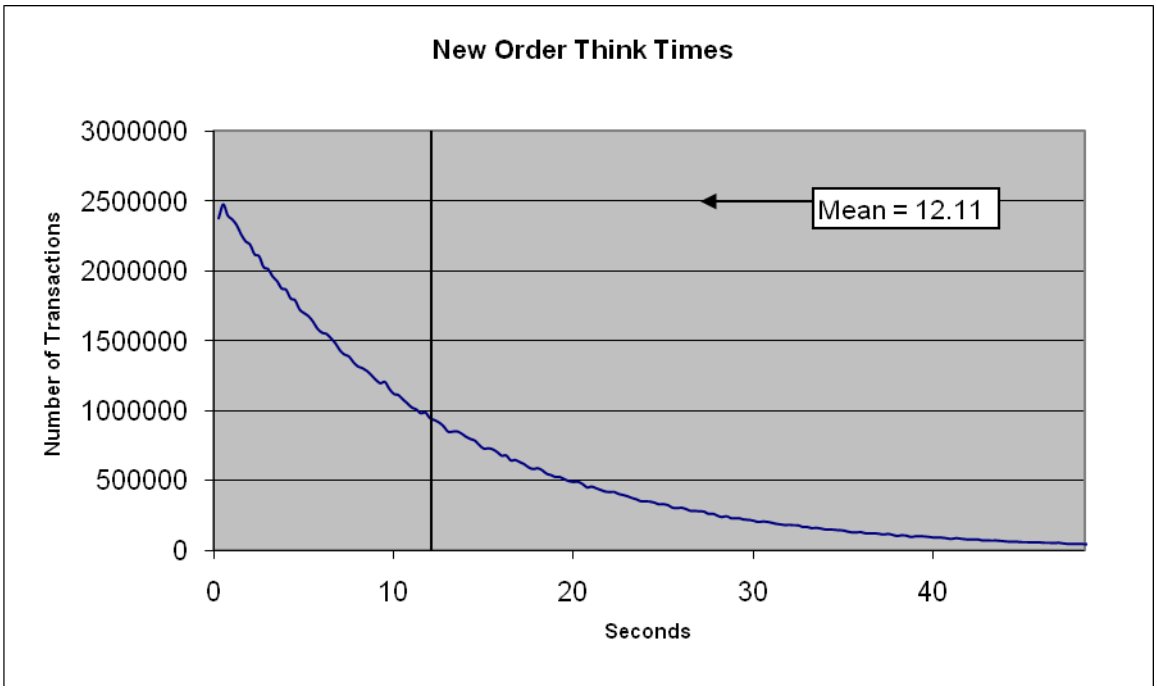
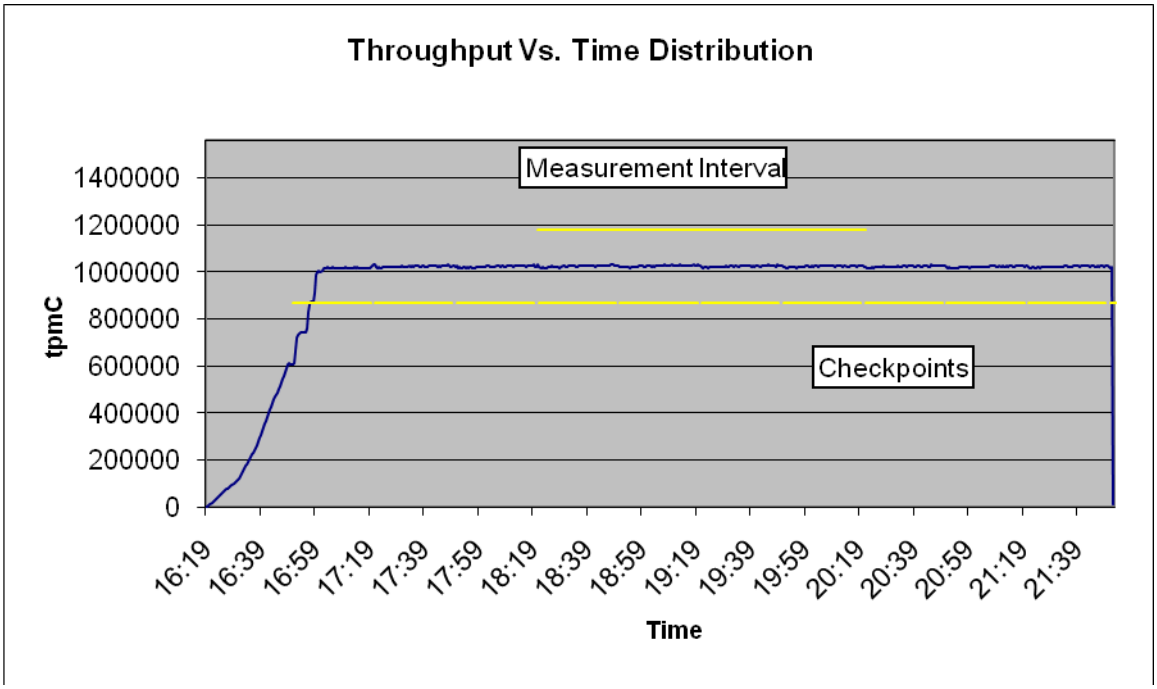


Figure 10. Throughput vs. Time Distribution



Steady State Determination

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

Steady state was determined using real time monitor utilities from the RTE. Steady state was further confirmed by the throughput data collected during the run and graphed in Figure 10.

Work Performed During Steady State

A description of how the work normally performed during a sustained test (for example check pointing, writing redo/undo log records, etc.), actually occurred during the measurement interval must be reported.

The RTE generated the required input data to choose a transaction from the menu. This data was time stamped. The input screen for the requested transaction was returned and time stamped. The difference between these two timestamps was the menu response time. The RTE writes to the log file once per transaction on selective fields such as order id. There is one log file per driver engine.

The RTE generated the required input data for the chosen transaction. It waited to complete the minimum required key time before transmitting the input screen. The transmission was time stamped. The return of the screen with the required response data was time stamped. The difference between these two timestamps was the response time for that transaction.

The RTE then waited the required think time interval before repeating the process starting at selecting a transaction from the menu.

The RTE transmissions were sent to application processes running on the client machines through Ethernet LANs. These client application processes handled all screen I/O as well as all requests to the database on the server. The applications communicated with the database server over gigabit Ethernet LANs using ODBC and RPC calls.

To perform checkpoints at specific intervals, the SQL Server *recovery interval* was set to 32767 and a script was written to schedule multiple checkpoints at specific intervals. The script included a wait time between each checkpoint equal to 30 minutes. The measurement interval was 120 minutes. The checkpoint script was started manually after the RTE had all users logged in and the database had achieved steady state.

At each checkpoint, Microsoft SQL Server wrote to disk all memory pages that had been updated but not yet physically written to disk. The positioning of the measurement interval is depicted on the graph in Figure 9.

Measurement Period Duration

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

The reported measured interval was exactly 120 minutes long.

Regulation of Transaction Mix

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

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

Transaction Statistics

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

Table 5.5: Transaction Statistics

Statistic		Value
New Order	Home warehouse order lines	99.00%
	Remote warehouse order lines	1.00%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home warehouse payments	85.00%
	Remote warehouse payments	15.00%
	Accessed by last name	59.96%
Delivery	Skipped transactions (interactive)	0
	Skipped transactions (deferred)	0
Order Status	Accessed by last name	59.95%
Transaction Mix	New Order	44.95%
	Payment	43.02%
	Order status	4.02%
	Delivery	4.00%
	Stock level	4.01%

Checkpoint Count and Location

The number of checkpoints in the Measurement Interval, the time in seconds from the start of the Measurement Interval to the first checkpoint, and the Checkpoint Interval must be disclosed.

The initial checkpoint was started 34 minutes after the start of the ramp-up. Subsequent checkpoints occurred every 30 minutes. Each checkpoint in the measurement interval lasted 29 minutes and 10 seconds. The measurement interval contains four checkpoints.

Checkpoint Duration

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

Checkpoint Start Time	Duration
4:51:41 PM	28 minutes, 20 seconds
5:21:38 PM	28 minutes, 20 seconds
5:51:35 PM	28 minutes, 20 seconds
6:21:32 PM	28 minutes, 20 seconds
6:51:29 PM	28 minutes, 20 seconds
7:21:26 PM	28 minutes, 20 seconds
7:51:23 PM	28 minutes, 20 seconds
8:21:20 PM	28 minutes, 20 seconds
8:51:17 PM	28 minutes, 20 seconds
9:21:14 PM	28 minutes, 20 seconds
9:51:11 PM	28 minutes, 20 seconds

Clause 6 Related Items

RTE Descriptions

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.

The RTE used was Microsoft Benchcraft RTE. Benchcraft is a proprietary tool provided by Microsoft and is not commercially available. The RTE's input is listed in Appendix A.

Emulated Components

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

The driver system consisted of 4 HP ProLiant servers hosting 8 Hyper-V virtual machines per system for a total of 32 VM client systems. These driver machines emulated the users' web browsers.

Functional Diagrams

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

The driver system performed the data generation and input functions of the priced display device. It also captured the input and output data and timestamps for post-processing of the reported metrics. No other functionality was included on the driver system.

Section 1.4 of this report contains detailed diagrams of both the benchmark configuration and the priced configuration.

Networks

The network configuration of both the tested services and proposed (target) services that 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.

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

In the tested configuration, 32 driver (RTE) machines were connected through a gigabit Ethernet switch to the client machines at 1Gbps, thus providing the path from the RTEs to the clients. The server (SUT) was connected to the clients through a gigabit Ethernet switch on a separate LAN.

The priced configuration was connected in the same manner as the tested configuration.

Operator Intervention

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

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

Clause 7 Related Items

System Pricing

A detailed list of hardware and software used in the priced system must be reported. Each separately orderable item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery data. If package-pricing is used, vendor part number of the package and a description uniquely identifying each of the components of the package must be disclosed. Pricing source and effective date(s) of price(s) must also be reported.

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

The details of the hardware and software are reported in the front of this report as part of the executive summary. All third party quotations are included at the end of this report as Appendix E.

Availability, Throughput, and Price Performance

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

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

- **Maximum Qualified Throughput** **1,024,380 tpmC**
- **Price per tpmC** **USD $0.65 per tpmC**
- **Availability** **June 20, 2011**

Country Specific Pricing

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

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

Usage Pricing

For any usage pricing, the sponsor must disclose:

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

The component pricing based on usage is shown below:

- 4 Microsoft Windows Server 2008 R2 Standard Edition
- 9 Microsoft Windows Server 2008 R2 Enterprise Edition
- 1 Microsoft SQL Server 2005 Enterprise x64 Edition (per processor) SP3
- 1 Microsoft Visual Studio 2008 Standard
- HP Servers include 3 years of support.

Clause 9 Related Items

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

This implementation of the TPC Benchmark C was audited by Lorna Livingtree of Performance Metrics, Inc.

Performance Metrics, Inc.
PO Box 984
Klamath CA 95548
(phone) 707-482-0523
(fax) 707-482-0575
e-mail: lornaL@perfmetrics.com

Availability of the Full Disclosure Report

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

Requests for this TPC Benchmark C Full Disclosure Report should be sent to:

TPC
Presidio of San Francisco
Building 572B Ruger St. (surface)
P.O. Box 29920 (mail)
San Francisco, CA 94129-0920

or

Hewlett-Packard Company
Performance Engineering
P.O. Box 692000
Houston, TX 77269-2000

Appendix A:

Source Code

The client source code is listed below.

dlldata.c

```
/*********************************************************
*****
   DllData file -- generated by MIDL compiler

        DO NOT ALTER THIS FILE

   This file is regenerated by MIDL on every IDL file
compile.

   To completely reconstruct this file, delete it and
rerun MIDL
   on all the IDL files in this DLL, specifying this
file for the
   /dlldata command line option

*********************************************************
****/


#include <rpcproxy.h>

#ifdef __cplusplus
extern "C"   {
#endif

EXTERN_PROXY_FILE( tpcc_com_ps )


PROXYFILE_LIST_START
/* Start of list */
  REFERENCE_PROXY_FILE( tpcc_com_ps ),
/* End of list */
PROXYFILE_LIST_END


DLLDATA_ROUTINES( aProxyFileList, GET_DLL_CLSID )

#ifdef __cplusplus
}  /*extern "C" */
#endif

/* end of generated dlldata file */
```

error.h

```
/*        FILE:               ERROR.H
 *                                      Microsoft
TPC-C Kit Ver. 4.69.000
 *                                      Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                                      Version
4.10.000 audited by Richard Gimarc, Performance
Metrics, 3/17/99
 *
 *        PURPOSE:  Header file for error exception
classes.
 *
 *  Change history:
 *                  4.20.000 - updated rev number to
match kit
 *                  4.21.000 - fixed bug: ~CBaseErr
needed to be declared virtual
 *                  4.69.000 - updated rev number to
match kit
 */

#pragma once

#ifndef _INC_STRING
          #include <string.h>
#endif

const int m_szMsg_size = 512;
const int m_szApp_size = 64;
const int m_szLoc_size = 64;

//error message structure used in ErrorText routines
typedef struct _SERRORMSG
{
          int                iError;
                    //error id of message
          char      szMsg[256];
          //message to sent to browser
} SERRORMSG;

typedef enum _ErrorLevel
{
          ERR_FATAL_LEVEL                          =
1,
          ERR_WARNING_LEVEL             = 2,
          ERR_INFORMATION_LEVEL         = 3
} ErrorLevel;


#define ERR_TYPE_LOGIC
                                        -1
          //logic error in program; internal error
#define ERR_SUCCESS
                                        0
          //success (a non-error error)
#define ERR_BAD_ITEM_ID
                                        1
          //expected abort record in txnRecord
#define ERR_TYPE_DELIVERY_POST
                                  2
          //expected delivery post failed
#define   ERR_TYPE_WEBDLL
                                        3
          //tpcc web generated error
#define   ERR_TYPE_SQL
                                        4
          //sql server generated error
#define   ERR_TYPE_DBLIB
                                        5
          //dblib generated error

#define   ERR_TYPE_ODBC
                                        6
          //odbc generated error
#define ERR_TYPE_SOCKET
                                        7
          //error on communication socket client rte
only
#define ERR_TYPE_DEADLOCK
                                  8
          //dblib and odbc only deadlock condition
#define ERR_TYPE_COM
                                        9
          //error from COM call
#define ERR_TYPE_TUXEDO
                                        10
          //tuxedo error
#define ERR_TYPE_OS
                                        11
          //operating system error
#define ERR_TYPE_MEMORY
                                        12
          //memory allocation error
#define ERR_TYPE_TPCC_ODBC
                                  13
          //error from tpcc odbc txn module
#define ERR_TYPE_TPCC_DBLIB
                                  14
          //error from tpcc dblib txn module
#define ERR_TYPE_DELISRV
                                  15
          //delivery server error
#define ERR_TYPE_TXNLOG
                                        16
          //txn log error
#define ERR_TYPE_BCCONN
                                        17
          //Benchcraft connection class
#define ERR_TYPE_TPCC_CONN
                                  18
          //Benchcraft connection class
#define ERR_TYPE_ENCINA
                                        19
          //Encina error
#define ERR_TYPE_COMPONENT
                                  20
          //error from COM component
#define ERR_TYPE_RTE
                                        21
          //Benchcraft rte
#define ERR_TYPE_AUTOMATION
                                  22
          //Benchcraft automation errors
#define ERR_TYPE_DRIVER
                                        23
          //Driver engine errors
#define ERR_TYPE_RTE_BASE
                                  24
          //Framework errors
#define   ERR_BUF_OVERFLOW
                                  25
          //Buffer overflow during receive
```

```
#define ERR_TYPE_SOAP_HTTP
                        26
          //HTTP/SOAP dll generated error
#define ERR_TYPE_OLEDB
                                  27
          //OLE-DB generated error
#define ERR_TYPE_TPCC_OLEDB
                        28
          //error from tpcc ole-db txn module
// TPC-W error types
#define ERR_TYPE_TPCW_CONN
                        50
          //Benchcraft connection class
#define ERR_TYPE_TPCW_HTML
                        51
          //error from TpcwHtml dll
#define ERR_TYPE_TPCW_USER
                        52
          //error from TPC-W user class
#define ERR_TYPE_TPCW_ENG_BASE
                        53
#define ERR_TYPE_TPCW_ENG_OS
                    54
#define ERR_TYPE_HTML_RESP
                        55
#define ERR_TYPE_TPCW_ODBC
                        56
#define ERR_TYPE_SCHANNEL
                        57
#define ERR_TYPE_THINK_LIST
                        58
//------ end TPC-W ----------
#define ERR_TYPE_XML_PROFILE
                    59
// TPC-E error types
#define ERR_TYPE_TPCE_CONN
                        60
          //TPC-E pipe connection errors
#define ERR_TYPE_TPCE_RTE
                        61
          //TPC-E Rte errors
#define ERR_TYPE_TPCE_ENG_BASE
                        62
          //Tpce Driver engine errors
#define ERR_TYPE_TPCE_ENG_OS
                    63                  //Tpce Driver
engine system errors
//#define ERR_TYPE_TPCE_MEE_ENG_BASE
                    64                  //Tpce MEE
Driver engine errors
//#define ERR_TYPE_TPCE_MEE_ENG_OS
                    65                  //Tpce MEE
Driver engine system errors

#define ERR_INS_MEMORY
          "Insufficient Memory to continue."
#define ERR_UNKNOWN
          "Unknown error."
#define ERR_MSG_BUF_SIZE                512
#define INV_ERROR_CODE                          -1
#define ERR_INS_BUF_OVERFLOW    "Insufficient Buffer
size to receive HTML pages."
```

```
class CBaseErr
{
public:
          enum Action
          {
                    eNone = 0
          };

          CBaseErr(LPCTSTR szLoc = NULL)
          {
                    m_idMsg             =
GetLastError();     //take the error code
immediatelly before it is reset by other functions


                    if (szLoc)
                    {
                              m_szLoc = new
char[strlen(szLoc)+1/*m_szLoc_size*/];
                              strcpy(m_szLoc, szLoc);
                    }
                    else
                              m_szLoc   = NULL;

                    m_szApp             = new
char[m_szApp_size];

          GetModuleFileName(GetModuleHandle(NULL),
m_szApp, m_szApp_size);
          }



          CBaseErr(int idMsg, LPCTSTR szLoc = NULL)
          {
                    m_idMsg             = idMsg;


                    if (szLoc)
                    {
                              m_szLoc = new
char[strlen(szLoc)+1/*m_szLoc_size*/];
                              strcpy(m_szLoc, szLoc);
                    }
                    else
                              m_szLoc   = NULL;

                    m_szApp             = new
char[m_szApp_size];

          GetModuleFileName(GetModuleHandle(NULL),
m_szApp, m_szApp_size);
          }

          virtual ~CBaseErr(void)
          {
                    if (m_szApp)
                              delete [] m_szApp;
                    if (m_szLoc)
                              delete [] m_szLoc;
          };

          virtual void Draw(HWND hwnd, LPCTSTR szStr
= NULL)
          {
                    int                 j = 0;
                    char      szTmp[512];

                    if (szStr)
                              j = wsprintf(szTmp,
"%s\n",szStr);
                    if (ErrorNum() != INV_ERROR_CODE)
                              j += wsprintf(szTmp+j,
"Error = %d\n", ErrorNum());
                    if (m_szLoc)
                              j += wsprintf(szTmp+j,
"Location = %s\n", GetLocation());

                    j += wsprintf(szTmp+j, "%s\n",
ErrorText());

                    MessageBox(hwnd, szTmp, m_szApp,
MB_OK);
          }

          char *GetApp(void) { return m_szApp; }
          char *GetLocation(void) { return m_szLoc; }
          virtual int ErrorNum() { return m_idMsg; }



          virtual int ErrorType() = 0;  // a value
which distinguishes the kind of error that occurred
          virtual char *ErrorTypeStr() = 0; // text
representation of the error type
          virtual char *ErrorText() = 0; // a string
(i.e., human readable) representation of the error
          virtual int ErrorAction() { return eNone; }
          // the function call that caused the error


protected:
          char      *m_szApp;
          char      *m_szLoc; // code location where
the error occurred
          int                 m_idMsg;

          //short   m_errType;
};

class CSocketErr : public CBaseErr
{
public:
          enum Action
          {
                    eNone = 0,
                    eSend,
                    eSocket,
                    eBind,
                    eConnect,
                    eListen,
                    eHost,
                    eRecv,
```

```
                    eGetHostByName,
                    eWSACreateEvent,
                    eWSASend,
                    eWSAGetOverlappedResult,
                    eWSARecv,
                    eWSAWaitForMultipleEvents,
                    eWSAStartup,
                    eWSAResetEvent,
                    eWSAEnumNetworkEvents,
                    eWSAEventSelect,
                    eSelect,
                    eAccept,
                    eNonRetryable
          };

          CSocketErr(Action eAction, LPCTSTR
szLocation = NULL);

          ~CSocketErr()
          {
                    if (m_szErrorText != NULL)
                              delete []
m_szErrorText;
          };

          Action    m_eAction;
          char      *m_szErrorText;

          int                 ErrorType() { return
ERR_TYPE_SOCKET;};
          char*     ErrorTypeStr() { return "SOCKET";
}
          char*     ErrorText(void);
          int                 ErrorAction() { return
(int)m_eAction; }

};

class CSystemErr : public CBaseErr
{
public:
          enum Action
          {
                    eNone = 0,
                    eTransactNamedPipe,
                    eWaitNamedPipe,
                    eSetNamedPipeHandleState,
                    eCreateFile,
                    eCreateProcess,
                    eCallNamedPipe,
                    eCreateEvent,
                    eCreateThread,
                    eVirtualAlloc,
                    eReadFile = 10,
                    eWriteFile,
                    eMapViewOfFile,
                    eCreateFileMapping,
                    eInitializeSecurityDescriptor,
                    eSetSecurityDescriptorDacl,
                    eCreateNamedPipe,
                    eConnectNamedPipe,
                    eWaitForSingleObject,
                    eRegOpenKeyEx,
                    eRegQueryValueEx = 20,
                    ebeginthread,
                    eRegEnumValue,
                    eRegSetValueEx,
                    eRegCreateKeyEx,
                    eWaitForMultipleObjects,
                    eRegisterClassEx,
                    eCreateWindow,
                    eCreateSemaphore,
                    eReleaseSemaphore,
                    eFSeek,
                    eFRead,
                    eFWrite,
                    eTmpFile,
                    eSetFilePointer,
                    eNew,
                    eCloseHandle,
                    eGetOverlappedResult
          };

                              CSystemErr(Action
eAction, LPCTSTR szLocation);
                              CSystemErr(int iError,
Action eAction, LPCTSTR szLocation);
          int                 ErrorType() { return
ERR_TYPE_OS;};
          char*     ErrorTypeStr() { return "SYSTEM";
}
          char      *ErrorText(void);
          int                 ErrorAction() { return
(int)m_eAction; }
          void      Draw(HWND hwnd, LPCTSTR szStr =
NULL);

          Action    m_eAction;

private:
          char m_szMsg[ERR_MSG_BUF_SIZE];
};

class CMemoryErr : public CBaseErr
{
public:
          CMemoryErr();

          int                 ErrorType() {return
ERR_TYPE_MEMORY;}
          char*     ErrorTypeStr() { return "OUT OF
MEMORY"; }
          char*     ErrorText() {return
ERR_INS_MEMORY;}
};


class CBufferOverflowErr : public CBaseErr
{
          public:
          CBufferOverflowErr(int,LPTSTR);

          int                 ErrorType() {return
ERR_BUF_OVERFLOW;}
          char*     ErrorTypeStr() { return "BUFFER
OVERFLOW"; }
          char*     ErrorText() {return
ERR_INS_BUF_OVERFLOW;}
};

// Exception type for XML profiles
class CXMLProfileErr : public CBaseErr
{
          public:
                    enum Action
                    {
                              LoadProfile = 1,
                              LoadSchema,
                              ValidateProfile,
                              SaveProfile,
                              LoadFromXML,
                              SaveToXML,

          ApplyProcessingInstruction,
                              ApplyAttribute,
                              ApplyNode
                    };

                    CXMLProfileErr(Action eAction,
int eCode, LPCTSTR szLocation)
                    {
                              m_eAction = eAction;
                              m_eCode = eCode;
                              m_bOverload = true;
                    };
                    CXMLProfileErr(Action eAction,
int eCode, LPCTSTR szLocation, char * szMsg)
                    {
                              m_eAction = eAction;
                              m_eCode = eCode;
                              strcpy(m_szMsg, szMsg);
                              m_bOverload = false;
                    };

                    virtual int
          ErrorType() { return
ERR_TYPE_XML_PROFILE;};
                    virtual char
          *ErrorTypeStr() { return "XML PROFILE"; };
                    virtual char        *ErrorText();

                    virtual int
          ErrorCode() { return m_eCode; };
                    int
          ErrorAction() { return (int)m_eAction; }
                    //virtual void      Draw(HWND
hwnd, LPCTSTR szStr = NULL)
                    //{
                    //        ::MessageBox(hwnd,
szStr, m_szLoc, MB_OK);
                    //};
          private:
                    char
          m_szMsg[ERR_MSG_BUF_SIZE];
                    LPCTSTR m_szLoc;
                    int                 m_eCode;
                    bool      m_bOverload;
                    Action    m_eAction;
```

```
};
```

install.c

```
/*        FILE:               INSTALL.C
 *                                    Microsoft
TPC-C Kit Ver. 4.69.000
 *                                    Copyright
Microsoft, 2008, 2009
 *               All Rights Reserved
 *
 *                                    not audited
 *
 *        PURPOSE:  Automated installation
application for TPC-C Web Kit
 *        Contact:  Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *                  4.20.000 - added COM installation
steps
 *      4.50.000 - added IIS6 configuration options
 *                  4.51.000 - added routines to copy
Visual Studio runtime module (MSVCR70.DLL)
 *                                          to
SystemRoot\System32
 *                  4.69.000 - added IIS7 suppport
and Windows Server 2008 R2 support
 */

#include <windows.h>
#include <direct.h>
#include <io.h>
#include <stdlib.h>
#include <tchar.h>
#include <stdio.h>
#include <commctrl.h>
#include "..\..\common\src\ReadRegistry.h"
#include <process.h>

#include "resource.h"

#define   WM_INITTEXT                   WM_USER+100

HICON               hIcon;
HINSTANCE hInst;

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

// TPC-C registry settings
TPCCREGISTRYDATA    Reg;

static    int                   iPoolThreadLimit;
static    int                   iMaxPoolThreads;
static    int                   iThreadTimeout;
static    int                   iListenBackLog;
static    int                   iAcceptExOutstanding;
static    int                   iUriEnableCache;
static    int                   iUriScavengerPeriod;
static    int                   iMaxConnections;

static    int                   iIISMajorVersion;
static    int                   iNumberOfProcessors;

static    int                   iMaxPhysicalMemory;
          //max physical memory in MB
static    char      szLastFileName[64];           //
last file we worked on (for error reporting)


BOOL      CALLBACK  LicenseDlgProc(HWND hwnd, UINT
uMsg, WPARAM wParam, LPARAM lParam);
BOOL      CALLBACK  UpdatedDlgProc(HWND hwnd, UINT
uMsg, WPARAM wParam, LPARAM lParam);
BOOL      CALLBACK  MainDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam);
BOOL      CALLBACK  CopyDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam);
static    void                  ProcessOK(HWND hwnd,
char *szDllPath, char *szWindowsPath);
static    void
          ReadRegistrySettings(void);
static    void
          WriteRegistrySettings(char *szDllPath);
static    BOOL                  RegisterDLL(char
*szFileName);
static    int
          CopyFiles(HWND hDlg, char *szDllPath, char
*szWindowsPath);
static    BOOL                  GetInstallPath(char
*szDllPath);
static    BOOL
          GetWindowsInstallPath(char *szWindowsPath);
static    void                  GetVersionInfo(char
*szDLLPath, char *szExePath);
static    BOOL
          CheckWWWebService(void);
static    BOOL
          StartWWWebService(void);
static    BOOL                  StopWWWebService(void);
static    void                  UpdateDialog(HWND
hDlg);
static    void                  ConfigureIIS6(HWND
hwnd, HWND hDlg);
static    void                  ConfigureIIS7(HWND
hwnd, HWND hDlg);

SYSTEM_INFO siSysInfo;

BOOL install_com(char *szDllPath);

#include "..\..\common\src\ReadRegistry.cpp"

int WINAPI WinMain( HINSTANCE hInstance, HINSTANCE
hPrevInstance, LPSTR lpCmdLine, int nCmdShow )
{
          int iRc;
          hInst = hInstance;

          InitCommonControls();

          hIcon = LoadIcon(hInstance,
MAKEINTRESOURCE(IDI_ICON1));

          iRc = DialogBox(hInstance,
MAKEINTRESOURCE(IDD_DIALOG4), GetDesktopWindow(),
LicenseDlgProc);
          if ( iRc )
          {
                    iRc = DialogBox(hInstance,
MAKEINTRESOURCE(IDD_DIALOG1), GetDesktopWindow(),
MainDlgProc);
                    if ( iRc )
                    {

          DialogBoxParam(hInstance,
MAKEINTRESOURCE(IDD_DIALOG2), GetDesktopWindow(),
UpdatedDlgProc, (LPARAM)iRc);
                    }
          }

          DestroyIcon(hIcon);
          return 0;
}

BOOL CALLBACK LicenseDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam)
{
          HGLOBAL                       hRes;
          HRSRC                         hResInfo;
          BYTE                          *pSrc, *pDst;
          DWORD                         dwSize;
          static HFONT        hFont;

          switch(uMsg)
          {
                    case WM_INITDIALOG:
                              hFont = CreateFont(-12,
0, 0, 0, 400, 0, 0, 0, 0, 0, 0, 0, 0, "Arial");
                              SendMessage(
GetDlgItem(hwnd, IDR_LICENSE1), WM_SETFONT,
(WPARAM)hFont, MAKELPARAM(0, 0) );
                              PostMessage(hwnd,
WM_INITTEXT, (WPARAM)0, (LPARAM)0);
                              return TRUE;
                    case WM_INITTEXT:
                              hResInfo =
FindResource(hInst, MAKEINTRESOURCE(IDR_LICENSE1),
"LICENSE");
                              dwSize =
SizeofResource(hInst, hResInfo);
                              hRes =
LoadResource(hInst, hResInfo );
                              pSrc = (BYTE
*)LockResource(hRes);
                              pDst = (unsigned char
*)malloc(dwSize+1);
                              if ( pDst )
                              {
```

```
                                        memcpy(pDst,
pSrc, dwSize);
                                        pDst[dwSize]
= 0;

          SetDlgItemText(hwnd, IDC_LICENSE, (const
char *)pDst);
                                        free(pDst);
                              }
                              else

          SetDlgItemText(hwnd, IDC_LICENSE, (const
char *)pSrc);
                              return TRUE;
                    case WM_DESTROY:
                              DeleteObject(hFont);
                              return TRUE;
                    case WM_COMMAND:
                              if ( wParam == IDOK )

          EndDialog(hwnd, TRUE);
                              if ( wParam == IDCANCEL
)

          EndDialog(hwnd, FALSE);
                    default:
                              break;
          }
          return FALSE;
}

BOOL CALLBACK UpdatedDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam)
{
          switch(uMsg)
          {
                    case WM_INITDIALOG:
                              switch(lParam)
                              {
                                        case 1:
                                        case 2:

          SetDlgItemText(hwnd, IDC_RESULTS, "TPC-C
Web Client Installed");

          break;
                              }
                              return TRUE;
                    case WM_COMMAND:
                              if ( wParam == IDOK )

          EndDialog(hwnd, TRUE);
                              break;
                    default:
                              break;
          }
          return FALSE;
}

BOOL CALLBACK MainDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam)
{
          PAINTSTRUCT                   ps;
          MEMORYSTATUS        memoryStatus;
          OSVERSIONINFO       VI;
          char
          szTmp[MAX_PATH];
          static char
          szDllPath[MAX_PATH];
          static char
          szWindowsPath[MAX_PATH];
          static char
          szExePath[MAX_PATH];

          switch(uMsg)
          {
                    case WM_INITDIALOG:

          GlobalMemoryStatus(&memoryStatus);
                              iMaxPhysicalMemory  =
(memoryStatus.dwTotalPhys/ 1048576);

                              if (
GetWindowsInstallPath(szWindowsPath) )
                              {

          MessageBox(hwnd, "Error: Cannot determine
Windows System Root.", NULL, MB_ICONSTOP | MB_OK);

          EndDialog(hwnd, FALSE);
                                        return TRUE;
                              }

                              if (
GetInstallPath(szDllPath) )
                              {

          MessageBox(hwnd, "Error internet service
inetsrv is not installed.", NULL, MB_ICONSTOP |
MB_OK);

          EndDialog(hwnd, FALSE);
                                        return TRUE;
                              }

                              // set default values
                              ZeroMemory( &Reg,
sizeof(Reg) );

          Reg.dwNumberOfDeliveryThreads = 4;
                              Reg.dwMaxConnections =
100;

          Reg.dwMaxPendingDeliveries    = 100;
                              Reg.eDB_Protocol =
ODBC;
                              Reg.eTxnMon = None;
                              strcpy(Reg.szDbServer,
                    "");
                              strcpy(Reg.szDbName,
                    "tpcc");
                              strcpy(Reg.szDbUser,
                    "sa");

          strcpy(Reg.szDbPassword,      "");
                              iPoolThreadLimit =
iMaxPhysicalMemory * 2;
                              iThreadTimeout = 86400;
                              iListenBackLog = 15;
                              iAcceptExOutstanding =
40;


          ReadTPCCRegistrySettings( &Reg );
                              ReadRegistrySettings();

                              // copy the hardware
information to the SYSTEM_INFO structure

          GetSystemInfo(&siSysInfo);
                              // store the number of
processors on this system
                              iNumberOfProcessors =
siSysInfo.dwNumberOfProcessors;


          GetModuleFileName(hInst, szExePath,
sizeof(szExePath));

          GetVersionInfo(szDllPath, szExePath);

                              wsprintf(szTmp,
"Version %d.%2.2d.%3.3d", versionExeMS, versionExeMM,
versionExeLS);
                              SetDlgItemText(hwnd,
IDC_VERSION, szTmp);

                              SetDlgItemText(hwnd,
IDC_PATH, szDllPath);

                              SetDlgItemText(hwnd,
ED_DB_SERVER, Reg.szDbServer);
                              SetDlgItemText(hwnd,
ED_DB_USER_ID, Reg.szDbUser);
                              SetDlgItemText(hwnd,
ED_DB_PASSWORD, Reg.szDbPassword);
                              SetDlgItemText(hwnd,
ED_DB_NAME, Reg.szDbName);

                              SetDlgItemInt(hwnd,
ED_THREADS, Reg.dwNumberOfDeliveryThreads, FALSE);
                              SetDlgItemInt(hwnd,
ED_MAXCONNECTION, Reg.dwMaxConnections, FALSE);
                              SetDlgItemInt(hwnd,
ED_MAXDELIVERIES, Reg.dwMaxPendingDeliveries, FALSE);
                              SetDlgItemInt(hwnd,
ED_IIS_MAX_THEAD_POOL_LIMIT, iPoolThreadLimit,
FALSE);
                              SetDlgItemInt(hwnd,
ED_IIS_THREAD_TIMEOUT, iThreadTimeout, FALSE);
                              SetDlgItemInt(hwnd,
ED_IIS_LISTEN_BACKLOG, iListenBackLog, FALSE);
                              SetDlgItemInt(hwnd,
ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE,
iAcceptExOutstanding, FALSE);

                              // check OS version
level for COM.  Must be at least Windows 2000
```

```
                                VI.dwOSVersionInfoSize
= sizeof(VI);
                                GetVersionEx( &VI );
                                if (VI.dwMajorVersion <
5)
                                {
                                        HWND hDlg =
GetDlgItem( hwnd, IDC_TM_MTS );
                                        EnableWindow(
hDlg, 0 );          // disable COM option
                                        if
(Reg.eTxnMon == COM)

          Reg.eTxnMon = None;
                                }

                                CheckDlgButton(hwnd,
IDC_TM_NONE,   0);
                                CheckDlgButton(hwnd,
IDC_TM_MTS,    0);
                                switch (Reg.eTxnMon)
                                {
                                case None:

          CheckDlgButton(hwnd, IDC_TM_NONE,   1);
                                        break;
                                case COM:

          CheckDlgButton(hwnd, IDC_TM_MTS,        1);
                                        break;
                                }

                                return TRUE;
                        case WM_PAINT:
                                if ( IsIconic(hwnd) )
                                {

          BeginPaint(hwnd, &ps);

          DrawIcon(ps.hdc, 0, 0, hIcon);

          EndPaint(hwnd, &ps);
                                        return TRUE;
                                }
                                break;
                        case WM_COMMAND:
                                if ( HIWORD(wParam) ==
BN_CLICKED )
                                {
                                        switch(
LOWORD(wParam) )
                                        {

          case IDOK:

          ProcessOK(hwnd, szDllPath, szWindowsPath);

          return TRUE;

          case IDCANCEL:

          EndDialog(hwnd, FALSE);

          return TRUE;

          default:

          return FALSE;
                                        }
                                }
                                break;
                        default:
                                break;
          }
          return FALSE;
}

static void ProcessOK(HWND hwnd, char *szDllPath,
char *szWindowsPath)
{
          int             d;
          HWND      hDlg;
          int             rc;
          BOOL      bSvcRunning;

          char      szFullName[MAX_PATH];
          char      szErrTxt[128];

          // Check whether Service Pack 1 has been
installed if
          // running on Windows Server 2003. The RTM
version has
          // a limitation on the number of concurrent
HTTP connections.
          //
          OSVERSIONINFOEX             VersionInfo;

          VersionInfo.dwOSVersionInfoSize =
sizeof(OSVERSIONINFOEX);
          if
(GetVersionEx((LPOSVERSIONINFO)&VersionInfo))
          {
                    if (VersionInfo.dwMajorVersion ==
5 &&      // Windows 2000/2003 Server?

          VersionInfo.dwMinorVersion == 2 &&      //
Windows 2003 Server?

          VersionInfo.wServicePackMajor == 0)     //
Service Pack installed?
                    {
                              TCHAR szMsg[MAX_PATH];


                              _sntprintf(szMsg,
sizeof(szMsg),
                                        "Warning:
running on Windows Server 2003 without at least
Service Pack 1\n"
                                        "limits the
number of concurrent HTTP connections to around
8000.");
                              MessageBox(hwnd, szMsg,
_T("Service Pack not Installed"), MB_ICONEXCLAMATION
| MB_OK);
                    }
          }

          // read settings from dialog
          Reg.dwNumberOfDeliveryThreads  =
GetDlgItemInt(hwnd, ED_THREADS, &d, FALSE);
          Reg.dwMaxConnections = GetDlgItemInt(hwnd,
ED_MAXCONNECTION, &d, FALSE);
          Reg.dwMaxPendingDeliveries =
GetDlgItemInt(hwnd, ED_MAXDELIVERIES, &d, FALSE);

          GetDlgItemText(hwnd, ED_DB_SERVER,
Reg.szDbServer, sizeof(Reg.szDbServer));
          GetDlgItemText(hwnd, ED_DB_USER_ID,
Reg.szDbUser, sizeof(Reg.szDbUser));
          GetDlgItemText(hwnd, ED_DB_PASSWORD,
Reg.szDbPassword, sizeof(Reg.szDbPassword));
          GetDlgItemText(hwnd, ED_DB_NAME,
Reg.szDbName, sizeof(Reg.szDbName));

          if ( IsDlgButtonChecked(hwnd, IDC_TM_NONE)
)
                    Reg.eTxnMon = None;
          else if ( IsDlgButtonChecked(hwnd,
IDC_TM_MTS) )
                    Reg.eTxnMon = COM;

          iPoolThreadLimit = GetDlgItemInt(hwnd,
ED_IIS_MAX_THEAD_POOL_LIMIT, &d, FALSE);
          iThreadTimeout = GetDlgItemInt(hwnd,
ED_IIS_THREAD_TIMEOUT,        &d, FALSE);
          iListenBackLog = GetDlgItemInt(hwnd,
ED_IIS_LISTEN_BACKLOG, &d, FALSE);
          iAcceptExOutstanding = GetDlgItemInt(hwnd,
ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE, &d, FALSE);


          ShowWindow(hwnd, SW_HIDE);
          hDlg = CreateDialog(hInst,
MAKEINTRESOURCE(IDD_DIALOG3), hwnd, CopyDlgProc);
          ShowWindow(hDlg, SW_SHOWNA);
          UpdateDialog(hDlg);

          // check to see if the web services are
running
          bSvcRunning = CheckWWWebService();
          if ( bSvcRunning )
          {
                    SetDlgItemText(hDlg, IDC_STATUS,
"Stopping Web Service.");
                    SendDlgItemMessage(hDlg,
IDC_PROGRESS1, PBM_STEPIT, 0, 0);
                    UpdateDialog(hDlg);

                    StopWWWebService();
                    SendDlgItemMessage(hDlg,
IDC_PROGRESS1, PBM_STEPIT, 0, 0);
                    UpdateDialog(hDlg);
          }

          // write binaries to inetpub\wwwroot
          rc = CopyFiles(hDlg, szDllPath,
szWindowsPath);
          if ( !rc )
```

```
        {
                ShowWindow(hwnd, SW_SHOWNA);
                DestroyWindow(hDlg);
                strcpy( szErrTxt, "Error(s)
occured when creating " );
                strcat( szErrTxt, szLastFileName
);
                MessageBox(hwnd, szErrTxt, NULL,
MB_ICONSTOP | MB_OK);
                EndDialog(hwnd, 0);
                return;
        }

        // while we have the web services shutdown,
check to see if this
        // is IIS6.  If it is, then call
ConfigureIIS6
        if ( iIISMajorVersion == 6)
        {
                ConfigureIIS6(hwnd, hDlg);
        }

        // while we have the web services shutdown,
check to see if this
        // is IIS7.  If it is, then call
ConfigureIIS6
        if ( iIISMajorVersion == 7)
        {
                ConfigureIIS7(hwnd, hDlg);
        }

        //if we stopped service restart it.
        if ( bSvcRunning )
        {
                SetDlgItemText(hDlg, IDC_STATUS,
"Starting Web Service.");
                SendDlgItemMessage(hDlg,
IDC_PROGRESS1, PBM_STEPIT, 0, 0);
                UpdateDialog(hDlg);
                StartWWWebService();
        }

        // update registry
        SetDlgItemText(hDlg, IDC_STATUS, "Updating
Registry.");
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);
        WriteRegistrySettings(szDllPath);

        // register com proxy stub
        strcpy(szFullName, szDllPath);
        strcat(szFullName, "tpcc_com_ps.dll");
        if (!RegisterDLL(szFullName))
        {
                ShowWindow(hwnd, SW_SHOWNA);
                DestroyWindow(hDlg);
                strcpy( szErrTxt, "Error occured
when registering " );
                strcat( szErrTxt, szFullName );
                MessageBox(hwnd, szErrTxt, NULL,
MB_ICONSTOP | MB_OK);
                EndDialog(hwnd, 0);
                return;
        }

        // if using COM
        if (Reg.eTxnMon == COM)
        {
                SetDlgItemText(hDlg, IDC_STATUS,
"Configuring COM.");
                SendDlgItemMessage(hDlg,
IDC_PROGRESS1, PBM_STEPIT, 0, 0);
                UpdateDialog(hDlg);

                if (install_com(szDllPath))
                {
                        ShowWindow(hwnd,
SW_SHOWNA);
                        DestroyWindow(hDlg);
                        strcpy( szErrTxt,
"Error occured when configuring COM settings." );
                        MessageBox(hwnd,
szErrTxt, NULL, MB_ICONSTOP | MB_OK);
                        EndDialog(hwnd, 0);
                        return;
                }
        }

        Sleep(100);

        ShowWindow(hwnd, SW_SHOWNA);
        DestroyWindow(hDlg);

        EndDialog(hwnd, rc);
        return;
}

static void ReadRegistrySettings(void)
{
        HKEY      hKey;
        DWORD     size;
        DWORD     type;

        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\InetStp", 0, KEY_READ, &hKey)
== ERROR_SUCCESS )
        {
                size = sizeof(iIISMajorVersion);
                if ( RegQueryValueEx(hKey,
"MajorVersion", 0, &type, (char *)&iIISMajorVersion,
&size) == ERROR_SUCCESS )
                        if ( !iIISMajorVersion
)

        iIISMajorVersion = 5;
        }

        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\Inetinfo\\Param
eters", 0, KEY_READ, &hKey) == ERROR_SUCCESS )
        {
                if ( iIISMajorVersion == 6)
                {
                        // since IIS6 handles
the pool thread parameters differently, we need to
fill in the dialog
                        // with the
MaxPoolThreads rather thatn PoolThreadLimit
                        // for ease of coding,
we are just going to stuff the value into
iPoolThreadLimit
                        size = sizeof(iPoolThreadLimit);
                        if (
RegQueryValueEx(hKey, "MaxPoolThreads", 0, &type,
(char *)&iPoolThreadLimit, &size) == ERROR_SUCCESS )
                        if ( !iPoolThreadLimit
)

        iPoolThreadLimit =  iMaxPhysicalMemory * 2;
                }
                else
                {
                        size =
sizeof(iPoolThreadLimit);
                        if (
RegQueryValueEx(hKey, "MaxPoolThreads", 0, &type,
(char *)&iPoolThreadLimit, &size) == ERROR_SUCCESS )
                        if ( !iPoolThreadLimit
)

        iPoolThreadLimit =  iMaxPhysicalMemory * 2;
                }

                size = sizeof(iThreadTimeout);
                if ( RegQueryValueEx(hKey,
"ThreadTimeout", 0, &type, (char *)&iThreadTimeout,
&size) == ERROR_SUCCESS )
                        if ( !iThreadTimeout )

        iThreadTimeout = 86400;

                size = sizeof(iListenBackLog);
                if ( RegQueryValueEx(hKey,
"ListenBackLog", 0, &type, (char *)&iListenBackLog,
&size) == ERROR_SUCCESS )
                        if ( !iListenBackLog )

        iListenBackLog = 15;

                RegCloseKey(hKey);
        }

        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Paramete
rs", 0, KEY_READ, &hKey) == ERROR_SUCCESS )
        {
                size =
sizeof(iAcceptExOutstanding);
                if ( RegQueryValueEx(hKey,
"AcceptExOutstanding", 0, &type, (char
*)&iAcceptExOutstanding, &size) == ERROR_SUCCESS )
                        if (
!iAcceptExOutstanding )

        iAcceptExOutstanding = 40;
```

```
                RegCloseKey(hKey);
        }
        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\HTTP\\Parameter
s", 0, KEY_READ, &hKey) == ERROR_SUCCESS )
        {
                size = sizeof(iUriEnableCache);
                if ( RegQueryValueEx(hKey,
"UriEnableCache", 0, &type, (char *)&iUriEnableCache,
&size) == ERROR_SUCCESS )
                        if ( !iUriEnableCache )

        iUriEnableCache = 0;

                size =
sizeof(iUriScavengerPeriod);
                if ( RegQueryValueEx(hKey,
"UriScavengerPeriod", 0, &type, (char
*)&iUriScavengerPeriod, &size) == ERROR_SUCCESS )
                        if (
!iUriScavengerPeriod )

        iUriScavengerPeriod = 10800;

                size = sizeof(iMaxConnections);
                if ( RegQueryValueEx(hKey,
"MaxConnections", 0, &type, (char *)&iMaxConnections,
&size) == ERROR_SUCCESS )
                        if ( !iMaxConnections )

        iMaxConnections = 100000;

                RegCloseKey(hKey);
        }
}


static void WriteRegistrySettings(char *szDllPath)
{
        HKEY    hKey;
        DWORD   dwDisposition;
        char    szTmp[MAX_PATH];
        char    *ptr;
        int             iRc;

        if ( RegCreateKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\TPCC", 0, NULL,
REG_OPTION_NON_VOLATILE, KEY_ALL_ACCESS, NULL, &hKey,
&dwDisposition) == ERROR_SUCCESS )
        {
                strcpy(szTmp, szDllPath);
                ptr = strstr(szTmp, "tpcc");
                if ( ptr )
                        *ptr = 0;

                RegSetValueEx(hKey, "Path", 0,
REG_SZ, szTmp, strlen(szTmp)+1);

                RegSetValueEx(hKey,
"NumberOfDeliveryThreads", 0, REG_DWORD, (char
*)&Reg.dwNumberOfDeliveryThreads,
sizeof(Reg.dwNumberOfDeliveryThreads));
```

```
                RegSetValueEx(hKey,
"MaxConnections", 0, REG_DWORD, (char
*)&Reg.dwMaxConnections,
sizeof(Reg.dwMaxConnections));
                RegSetValueEx(hKey,
"MaxPendingDeliveries", 0, REG_DWORD, (char
*)&Reg.dwMaxPendingDeliveries,
sizeof(Reg.dwMaxPendingDeliveries));

                RegSetValueEx(hKey,
"DB_Protocol", 0, REG_SZ,
szDBNames[Reg.eDB_Protocol],
strlen(szDBNames[Reg.eDB_Protocol])+1);
                RegSetValueEx(hKey, "TxnMonitor",
0, REG_SZ, szTxnMonNames[Reg.eTxnMon],
strlen(szTxnMonNames[Reg.eTxnMon])+1);

                RegSetValueEx(hKey, "DbServer",
0, REG_SZ, Reg.szDbServer, strlen(Reg.szDbServer)+1);
                RegSetValueEx(hKey, "DbName", 0,
REG_SZ, Reg.szDbName, strlen(Reg.szDbName)+1);
                RegSetValueEx(hKey, "DbUser", 0,
REG_SZ, Reg.szDbUser, strlen(Reg.szDbUser)+1);
                RegSetValueEx(hKey, "DbPassword",
0, REG_SZ, Reg.szDbPassword,
strlen(Reg.szDbPassword)+1);

                strcpy(szTmp, "YES");
                RegSetValueEx(hKey,
"COM_SinglePool", 0, REG_SZ, szTmp, strlen(szTmp)+1);

                RegFlushKey(hKey);
                RegCloseKey(hKey);
        }

        if (
(iRc=RegCreateKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\Inetinfo\\Param
eters", 0, NULL, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, NULL, &hKey, &dwDisposition)) ==
ERROR_SUCCESS )
        {
                // if this is IIS6, then we need
to treat the PoolThreadLimit differently
                // if IIS6, then PoolThreadLimit
is the maximum number of threads for the entire
system.
                // IIS6 added MaxPoolThreads
which controls the number of threads per processor.
For IIS6
                // we will set MaxPoolThreads to
the value the user proivided in the dialog and then
set
                // PoolThreadLimit to
MaxPoolThreads * number of processors on this system
                if ( iIISMajorVersion == 6 )
                {
                        iMaxPoolThreads =
iPoolThreadLimit;
                        iPoolThreadLimit =
iMaxPoolThreads * iNumberOfProcessors;
```

```
                        RegSetValueEx(hKey,
"PoolThreadLimit", 0, REG_DWORD, (char
*)&iPoolThreadLimit, sizeof(iPoolThreadLimit));
                        RegSetValueEx(hKey,
"MaxPoolThreads", 0, REG_DWORD, (char
*)&iMaxPoolThreads, sizeof(iMaxPoolThreads));
                }
                else
                {
                        RegSetValueEx(hKey,
"PoolThreadLimit", 0, REG_DWORD, (char
*)&iPoolThreadLimit, sizeof(iPoolThreadLimit));
                }

                RegSetValueEx(hKey,
"ThreadTimeout", 0, REG_DWORD, (char
*)&iThreadTimeout, sizeof(iThreadTimeout));
                RegSetValueEx(hKey,
"ListenBackLog", 0, REG_DWORD, (char
*)&iListenBackLog, sizeof(iListenBackLog));

                RegFlushKey(hKey);
                RegCloseKey(hKey);
        }

        if (
(iRc=RegCreateKeyEx(HKEY_LOCAL_MACHINE,
"SYSTEM\\CurrentControlSet\\Services\\W3SVC\\Paramete
rs", 0, NULL, REG_OPTION_NON_VOLATILE,
KEY_ALL_ACCESS, NULL, &hKey, &dwDisposition)) ==
ERROR_SUCCESS )
        {
                RegSetValueEx(hKey,
"AcceptExOutstanding", 0, REG_DWORD, (char
*)&iAcceptExOutstanding,
sizeof(iAcceptExOutstanding));

                RegFlushKey(hKey);
                RegCloseKey(hKey);
        }

        return;
}

BOOL CALLBACK CopyDlgProc(HWND hwnd, UINT uMsg,
WPARAM wParam, LPARAM lParam)
{
        if ( uMsg == WM_INITDIALOG )
        {
                SendDlgItemMessage(hwnd,
IDC_PROGRESS1, PBM_SETRANGE, 0, MAKELPARAM(0, 13));
                SendDlgItemMessage(hwnd,
IDC_PROGRESS1, PBM_SETSTEP, (WPARAM)1, 0);
                return TRUE;
        }
        return FALSE;
}

BOOL RegisterDLL(char *szFileName)
{
        HINSTANCE hLib;
        FARPROC             lpDllEntryPoint;
```

```
        hLib = LoadLibrary(szFileName);
        if ( hLib == NULL )
                return FALSE;
        // Find the entry point.
        lpDllEntryPoint = GetProcAddress(hLib,
"DllRegisterServer");
        if (lpDllEntryPoint != NULL)
        {
                return ((*lpDllEntryPoint)() ==
S_OK);
        }
        else
                return FALSE;       //unable to
locate entry point
}

BOOL FileFromResource( char *szResourceName, int
iResourceId, char *szDllPath, char *szFileName )
{
        HGLOBAL                         hDLL;
        HRSRC                           hResInfo;
        HANDLE                          hFile;
        DWORD                           dwSize;
        BYTE                            *pSrc;
        DWORD                           d;
        char
        szFullName[MAX_PATH];

        hResInfo = FindResource(hInst,
MAKEINTRESOURCE(iResourceId), szResourceName);

        strcpy(szFullName, szDllPath);
        strcat(szFullName, szFileName);

        dwSize = SizeofResource(hInst, hResInfo);
        hDLL = LoadResource(hInst, hResInfo );
        pSrc = (BYTE *)LockResource(hDLL);
        //remove(szFullName);

        hFile = CreateFile(szFullName,
GENERIC_WRITE, 0, NULL, CREATE_ALWAYS,
FILE_ATTRIBUTE_NORMAL, NULL);
        if (hFile == INVALID_HANDLE_VALUE)
        {
                DWORD dwError = GetLastError();
                return FALSE;
        }

        if ( !WriteFile(hFile, pSrc, dwSize, &d,
NULL) )
                return FALSE;

        CloseHandle(hFile);

        UnlockResource(hDLL);
        FreeResource(hDLL);
        return TRUE;
}

static int CopyFiles(HWND hDlg, char *szDllPath, char
*szWindowsPath)
{
        SetDlgItemText(hDlg, IDC_STATUS, "Copying
Files...");
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        // install TPCC.DLL
        strcpy( szLastFileName, "tpcc.dll" );
        if (!FileFromResource( "TPCCDLL",
IDR_TPCCDLL, szDllPath, szLastFileName ))
                return 0;
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);


        // install MSVCR71.DLL
        strcpy( szLastFileName, "msvcr71.dll" );
        if (!FileFromResource( "MSVCR71",
IDR_MSVCR71, szWindowsPath, szLastFileName ))
                return 0;
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        // install tpcc_odbc.dll
        strcpy( szLastFileName, "tpcc_odbc.dll" );
        if (!FileFromResource( "ODBC_DLL",
IDR_ODBC_DLL, szDllPath, szLastFileName ))
                return 0;
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        // install tpcc_com.dll
        strcpy( szLastFileName, "tpcc_com.dll" );
        if (!FileFromResource( "COM_DLL",
IDR_COM_DLL, szDllPath, szLastFileName ))
                return 0;
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        // install tpcc_com_all.tlb
        strcpy( szLastFileName, "tpcc_com_all.tlb"
);
        if (!FileFromResource( "COM_TYPLIB",
IDR_COMTYPLIB_DLL, szDllPath, szLastFileName ))
                return 0;
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        // install tpcc_com_ps.dll
        strcpy( szLastFileName, "tpcc_com_ps.dll"
);
        if (!FileFromResource( "COM_PS_DLL",
IDR_COMPS_DLL, szDllPath, szLastFileName ))
                return 0;
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        // install tpcc_com_all.dll
        strcpy( szLastFileName, "tpcc_com_all.dll"
);
        if (!FileFromResource( "COM_ALL_DLL",
IDR_COMALL_DLL, szDllPath, szLastFileName ))
                return 0;
        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        return 1;
}

static BOOL GetInstallPath(char *szDllPath)
{
    HKEY  hKey;
        BYTE      szData[MAX_PATH];
        DWORD     sv;
        BOOL      bRc;
        int                 len;
        int                 iRc;


        // Registry key
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\InetStp\PathWWW
Root is used to find the
        // IIS default web site directory and
determine that IIS is installed.

        szDllPath[0] =  0;
        bRc = TRUE;
        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\InetStp", 0, KEY_ALL_ACCESS,
&hKey) == ERROR_SUCCESS )
        {
                sv = sizeof(szData);
                iRc = RegQueryValueEx( hKey,
"PathWWWRoot", NULL, NULL, szData, &sv );  // used by
IIS 5.0 & 6.0
                if (iRc == ERROR_SUCCESS)
                {
                        len =
ExpandEnvironmentStrings(szData, szDllPath,
MAX_PATH);
                        if (len < MAX_PATH)
                        {
                                if (
szDllPath[len-2] != '\\' )
                                {

        szDllPath[len-1] = '\\';

        szDllPath[len] = 0;
                                }
                                bRc = FALSE;
                        }
                }

                RegCloseKey(hKey);
```

```
        }

        return bRc;
}

static BOOL GetWindowsInstallPath(char
*szWindowsPath)
{
    HKEY  hKey;
        BYTE      szData[MAX_PATH];
        DWORD     sv;
        BOOL      bRc;
        int                 len;
        int                 iRc;


        // Registry key
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Windows
NT\CurrentVersion\SystemRoot is used to find the
        // system root to install the VC70 DLL.

        szWindowsPath[0] =  0;
        bRc = TRUE;
        if ( RegOpenKeyEx(HKEY_LOCAL_MACHINE,
"SOFTWARE\\Microsoft\\Windows NT\\CurrentVersion", 0,
KEY_ALL_ACCESS, &hKey) == ERROR_SUCCESS )
        {
                sv = sizeof(szData);
                iRc = RegQueryValueEx( hKey,
"SystemRoot", NULL, NULL, szData, &sv );
                if (iRc == ERROR_SUCCESS)
                {
                        bRc = FALSE;
                        strcpy(szWindowsPath,
szData);
                        len =
strlen(szWindowsPath);
                        if ( szWindowsPath[len-
1] != '\\' )
                        {

        szWindowsPath[len] = '\\';

        szWindowsPath[len+1] = 0;
                        }
                        // now append the path
to SYSTEM32
                        strcat(szWindowsPath,
"SYSTEM32\\");
                }

                RegCloseKey(hKey);
        }

        return bRc;
}
static void GetVersionInfo(char *szDLLPath, char
*szExePath)
{
        DWORD                                   d;
        DWORD
        dwSize;
        DWORD
        dwBytes;
        char
        *ptr;
        VS_FIXEDFILEINFO    *vs;

        versionDllMS = 0;
        versionDllLS = 0;
        if ( _access(szDLLPath, 00) == 0 )
        {
                dwSize =
GetFileVersionInfoSize(szDLLPath, &d);
                if ( dwSize )
                {
                        ptr = (char
*)malloc(dwSize);

        GetFileVersionInfo(szDLLPath, 0, dwSize,
ptr);
                        VerQueryValue(ptr,
"\\",&vs, &dwBytes);
                        versionDllMS = vs-
>dwProductVersionMS;
                        versionDllLS = vs-
>dwProductVersionLS;
                        free(ptr);
                }
        }

        versionExeMS = 0x7FFF;
        versionExeLS = 0x7FFF;
        dwSize = GetFileVersionInfoSize(szExePath,
&d);
        if ( dwSize )
        {
                ptr = (char *)malloc(dwSize);
                GetFileVersionInfo(szExePath, 0,
dwSize, ptr);
                VerQueryValue(ptr, "\\",&vs,
&dwBytes);

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

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

        schSCManager = OpenSCManager(NULL, NULL,
SC_MANAGER_ALL_ACCESS);
        schService = OpenService(schSCManager,
TEXT("W3SVC"), SERVICE_ALL_ACCESS);
        if (schService == NULL)
                return FALSE;

        if (! QueryServiceStatus(schService,
&ssStatus) )
                goto ServiceNotRunning;

        if ( !ControlService(schService,
SERVICE_CONTROL_STOP, &ssStatus) )
                goto ServiceNotRunning;
        //start Service pending, Check the status
until the service is running.
        if (! QueryServiceStatus(schService,
&ssStatus) )
                goto ServiceNotRunning;

        CloseServiceHandle(schService);
        return TRUE;

ServiceNotRunning:

        CloseServiceHandle(schService);
        return FALSE;
}

static BOOL StartWWWebService(void)
{
        SC_HANDLE           schSCManager;
        SC_HANDLE           schService;
        SERVICE_STATUS      ssStatus;
        DWORD
        dwOldCheckPoint;

        schSCManager = OpenSCManager(NULL, NULL,
SC_MANAGER_ALL_ACCESS);
        schService = OpenService(schSCManager,
TEXT("W3SVC"), SERVICE_ALL_ACCESS);
        if (schService == NULL)
                return FALSE;

        if (! StartService(schService, 0, NULL) )
                goto StartWWWebErr;
        //start Service pending, Check the status
until the service is running.
        if (! QueryServiceStatus(schService,
&ssStatus) )
                goto StartWWWebErr;
        while( ssStatus.dwCurrentState !=
SERVICE_RUNNING)
        {

                dwOldCheckPoint =
ssStatus.dwCheckPoint;
        //Save the current checkpoint.
                Sleep(ssStatus.dwWaitHint);

        //Wait for the specified interval.
                if (
!QueryServiceStatus(schService, &ssStatus) )
        //Check the status again.
                        break;
                if (dwOldCheckPoint >=
ssStatus.dwCheckPoint)                  //Break if
the checkpoint has not been incremented.
```

```
                        break;
        }

        if (ssStatus.dwCurrentState ==
SERVICE_RUNNING)
                goto StartWWWebErr;

        CloseServiceHandle(schService);
        return TRUE;

StartWWWebErr:
        CloseServiceHandle(schService);
        return FALSE;
}

static BOOL StopWWWebService(void)
{
        SC_HANDLE           schSCManager;
        SC_HANDLE           schService;
        SERVICE_STATUS      ssStatus;
        DWORD
        dwOldCheckPoint;

        schSCManager = OpenSCManager(NULL, NULL,
SC_MANAGER_ALL_ACCESS);
        //schService = OpenService(schSCManager,
TEXT("W3SVC"), SERVICE_ALL_ACCESS);
        schService = OpenService(schSCManager,
TEXT("IISADMIN"), SERVICE_ALL_ACCESS);
        if (schService == NULL)
                return FALSE;

        if (! QueryServiceStatus(schService,
&ssStatus) )
                goto StopWWWebErr;

        if ( !ControlService(schService,
SERVICE_CONTROL_STOP, &ssStatus) )
                goto StopWWWebErr;
        //start Service pending, Check the status
until the service is running.
        if (! QueryServiceStatus(schService,
&ssStatus) )
                goto StopWWWebErr;
        while( ssStatus.dwCurrentState ==
SERVICE_RUNNING)
        {

                dwOldCheckPoint =
ssStatus.dwCheckPoint;
        //Save the current checkpoint.
                Sleep(ssStatus.dwWaitHint);

        //Wait for the specified interval.
                if (
!QueryServiceStatus(schService, &ssStatus) )
        //Check the status again.
                        break;
                if (dwOldCheckPoint >=
ssStatus.dwCheckPoint)                  //Break if
the checkpoint has not been incremented.
                        break;
        }

        if (ssStatus.dwCurrentState ==
SERVICE_RUNNING)
                goto StopWWWebErr;

        CloseServiceHandle(schService);
        return TRUE;

StopWWWebErr:
        CloseServiceHandle(schService);
        return FALSE;
}

static void UpdateDialog(HWND hDlg)
{
        MSG msg;

        UpdateWindow(hDlg);
        while( PeekMessage(&msg, hDlg, 0, 0,
PM_REMOVE) )
        {
                TranslateMessage(&msg);
                DispatchMessage(&msg);
        }
        Sleep(250);
        return;
}

static void ConfigureIIS6(HWND hwnd, HWND hDlg)
{
        int                 irc;
        char      szErrTxt[128];
        FILE      *fErrorFile;

        SetDlgItemText(hDlg, IDC_STATUS,
"Configuring IIS6...");
        //SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        irc = system("IIS6_CONFIG.CMD");

        // since the return code from the command
file is always 1,
        // check to see if the file iis6_config.err
exists
        // if it does, then something hosed
        fErrorFile = fopen("IIS6_CONFIG.err","r");
        if ( fErrorFile != NULL )
        {
                ShowWindow(hwnd, SW_SHOWNA);
                DestroyWindow(hDlg);
                strcpy( szErrTxt, "IIS6
configuration error." );
                strcat( szErrTxt, "Check
iis6_config.err" );
                MessageBox(hwnd, szErrTxt, NULL,
MB_ICONSTOP | MB_OK);
                EndDialog(hwnd, 0);
                return;
        }
}

static void ConfigureIIS7(HWND hwnd, HWND hDlg)
{
        int                 irc;
        char      szErrTxt[128];
        FILE      *fErrorFile;

        SetDlgItemText(hDlg, IDC_STATUS,
"Installing VS Modules...");
        UpdateDialog(hDlg);

        if ( access( "%SystemRoot%\\System32", 0)
== 0 )
        {

        CopyFile("..\\VS_Modules\\ATL71.DLL",
"%SystemRoot%\\System32", 0);

        CopyFile("..\\VS_Modules\\MSVCR71D.DLL",
"%SystemRoot%\\System32", 0);

        CopyFile("..\\VS_Modules\\MSVCP71D.DLL",
"%SystemRoot%\\System32", 0);
        }

        if ( access( "%SystemRoot%\\SysWOW64", 0)
== 0 )
        {

        CopyFile("..\\VS_Modules\\ATL71.DLL",
"%SystemRoot%\\SysWOW64", 0);

        CopyFile("..\\VS_Modules\\MSVCR71D.DLL",
"%SystemRoot%\\SysWOW64", 0);

        CopyFile("..\\VS_Modules\\MSVCP71D.DLL",
"%SystemRoot%\\SysWOW64", 0);
        }

        SetDlgItemText(hDlg, IDC_STATUS,
"Configuring IIS7...");
        //SendDlgItemMessage(hDlg, IDC_PROGRESS1,
PBM_STEPIT, 0, 0);
        UpdateDialog(hDlg);

        irc = system("IIS7_CONFIG.CMD");

        // since the return code from the command
file is always 1,
        // check to see if the file iis6_config.err
exists
        // if it does, then something hosed
        fErrorFile = fopen("IIS7_CONFIG.err","r");
        if ( fErrorFile != NULL )
        {
                ShowWindow(hwnd, SW_SHOWNA);
                DestroyWindow(hDlg);
                strcpy( szErrTxt, "IIS7
configuration error." );
                strcat( szErrTxt, "Check
iis7_config.err" );
                MessageBox(hwnd, szErrTxt, NULL,
MB_ICONSTOP | MB_OK);
                EndDialog(hwnd, 0);
```

```
                return;
        }
}
```

install.h

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

#define IDD_DIALOG1                             101
#define IDI_ICON1                               102
#define IDR_TPCCDLL                             103
#define IDD_DIALOG2                             105
#define IDI_ICON2                               106
#define IDR_DELIVERY                            107
#define IDD_DIALOG3                             108

#define BN_LOG
          1001
#define ED_KEEP
          1002
#define ED_THREADS
          1003
#define ED_THREADS2
          1004
#define IDC_PATH
          1007
#define IDC_VERSION
          1009
#define IDC_RESULTS
          1010
#define IDC_PROGRESS1
          1011
#define IDC_STATUS
          1012
#define IDC_BUTTON1
          1013
#define ED_MAXCONNECTION
          1014
#define ED_IIS_MAX_THEAD_POOL_LIMIT
          1015
#define ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE
          1017
#define ED_IIS_THREAD_TIMEOUT
          1018
#define ED_IIS_LISTEN_BACKLOG
          1019
#define IDC_ODBC
          1022
#define IDC_CONNECT_POOL
                  1023
#define ED_USER_CONNECT_DELAY_TIME
          1024

// Next default values for new objects
//
```

install_com.cpp

```
/*        FILE:               INSTALL_COM.CPP
 *                                          Microsoft
TPC-C Kit Ver. 4.69.000
 *                                          Copyright
Microsoft, 2008, 2009
 *             All Rights Reserved
 *
 *                                          not audited
 *
 *        PURPOSE:  installation code for COM
application for TPC-C Web Kit
 *        Contact:  Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *                  4.20.000 - first version
 */

#define _WIN32_WINNT 0x0500

#include <comdef.h>
#include <comadmin.h>
#include <stdio.h>
#include <tchar.h>

extern "C"
{
          BOOL install_com(char *szDllPath);
}

BOOL install_com(char *szDllPath)
{
          ICOMAdminCatalog*   pCOMAdminCat = NULL;
          ICatalogCollection* pCatalogCollectionApp
= NULL;
          ICatalogCollection* pCatalogCollectionCo
= NULL;
          ICatalogCollection* pCatalogCollectionItf
= NULL;
          ICatalogCollection*
          pCatalogCollectionMethod = NULL;

          ICatalogObject*
          pCatalogObjectApp    = NULL;
          ICatalogObject*
          pCatalogObjectCo     = NULL;
          ICatalogObject*
          pCatalogObjectItf    = NULL;
          ICatalogObject*
          pCatalogObjectMethod = NULL;

          _bstr_t
          bstrTemp, bstrTemp2, bstrTemp3, bstrTemp4;
          _bstr_t
          bstrDllPath = szDllPath;
          _variant_t
          vTmp, vKey;
          long
          lActProp, lCount, lCountCo, lCountItf,
lCountMethod;
          bool
          bTmp;


          CoInitializeEx(NULL, COINIT_MULTITHREADED);

          HRESULT hr =
CoCreateInstance(CLSID_COMAdminCatalog,

                                          NULL,


          CLSCTX_INPROC_SERVER,


          IID_ICOMAdminCatalog,

                                          (void**)
&pCOMAdminCat);

          if (!SUCCEEDED(hr)) goto Error;

          bstrTemp = "Applications";

          // Attempt to connect to "Applications" in
the Catalog
          hr = pCOMAdminCat->GetCollection(bstrTemp,

                                          (IDispatch**)
&pCatalogCollectionApp);
          if (!SUCCEEDED(hr)) goto Error;

          // Attempt to load the "Applications"
collection
          hr = pCatalogCollectionApp->Populate();
          if (!SUCCEEDED(hr)) goto Error;

          hr = pCatalogCollectionApp-
>get_Count(&lCount);
          if (!SUCCEEDED(hr)) goto Error;

          // iterate through applications to delete
existing "TPC-C" application (if any)
          while (lCount > 0)
          {
                    hr = pCatalogCollectionApp-
>get_Item(lCount - 1, (IDispatch**)
&pCatalogObjectApp);
                    if (!SUCCEEDED(hr)) goto Error;

                    hr = pCatalogObjectApp-
>get_Name(&vTmp);
                    if (!SUCCEEDED(hr)) goto Error;

                    if (wcscmp(vTmp.bstrVal, L"TPC-
C"))
                    {
                              lCount--;
                              continue;
                    }
```

```
                    else
                    {
                              hr =
pCatalogCollectionApp->Remove(lCount - 1);
                              if (!SUCCEEDED(hr))
goto Error;
                              break;
                    }
          }

          hr = pCatalogCollectionApp-
>SaveChanges(&lActProp);
          if (!SUCCEEDED(hr)) goto Error;

          // add the new application
          hr = pCatalogCollectionApp-
>Add((IDispatch**) &pCatalogObjectApp);
          if (!SUCCEEDED(hr)) goto Error;

          // set properties
          bstrTemp = "Name";
          vTmp = "TPC-C";
          hr = pCatalogObjectApp->put_Value(bstrTemp,
vTmp);
          if (!SUCCEEDED(hr)) goto Error;

          // set as a library (in process)
application
          bstrTemp = "Activation";
          lActProp = COMAdminActivationInproc;
          vTmp = lActProp;
          hr = pCatalogObjectApp->put_Value(bstrTemp,
vTmp);
          if (!SUCCEEDED(hr)) goto Error;

          // set security level to process
          bstrTemp = "AccessChecksLevel";
          lActProp =
COMAdminAccessChecksApplicationLevel;
          vTmp = lActProp;
          hr = pCatalogObjectApp->put_Value(bstrTemp,
vTmp);
          if (!SUCCEEDED(hr)) goto Error;

          // save key to get the Components
collection later
          hr = pCatalogObjectApp->get_Key(&vKey);
          if (!SUCCEEDED(hr)) goto Error;

          // save changes (app creation) so component
installation will work
          hr = pCatalogCollectionApp-
>SaveChanges(&lActProp);
          if (!SUCCEEDED(hr)) goto Error;

          pCatalogObjectApp->Release();
          pCatalogObjectApp = NULL;

          bstrTemp  = "TPC-C";
                    // app name
          bstrTemp2 =         bstrDllPath +
"tpcc_com_all.dll";           // DLL
          bstrTemp3 =         bstrDllPath +
"tpcc_com_all.tlb";           // type library (TLB)
          bstrTemp4 =         bstrDllPath +
"tpcc_com_ps.dll";            // proxy/stub dll

          hr = pCOMAdminCat-
>InstallComponent(bstrTemp,


          bstrTemp2,


          bstrTemp3,


          bstrTemp4);
          if (!SUCCEEDED(hr)) goto Error;


          bstrTemp = "Components";
          hr = pCatalogCollectionApp-
>GetCollection(bstrTemp, vKey, (IDispatch**)
&pCatalogCollectionCo);
          if (!SUCCEEDED(hr)) goto Error;

          hr = pCatalogCollectionCo->Populate();
          if (!SUCCEEDED(hr)) goto Error;

          hr = pCatalogCollectionCo-
>get_Count(&lCountCo);
          if (!SUCCEEDED(hr)) goto Error;

          // iterate through components in
application and set the properties
          while (lCountCo > 0)
          {
                    hr = pCatalogCollectionCo-
>get_Item(lCountCo - 1, (IDispatch**)
&pCatalogObjectCo);
                    if (!SUCCEEDED(hr)) goto Error;

                    // used for debugging (view the
name)
                    hr = pCatalogObjectCo-
>get_Name(&vTmp);
                    if (!SUCCEEDED(hr)) goto Error;

                    bstrTemp = "ConstructionEnabled";
                    bTmp = TRUE;
                    vTmp = bTmp;
                    hr = pCatalogObjectCo-
>put_Value(bstrTemp, vTmp);
                    if (!SUCCEEDED(hr)) goto Error;

                    bstrTemp = "ConstructorString";
                    bstrTemp2 = "dummy string (do not
remove)";
                    vTmp = bstrTemp2;
                    hr = pCatalogObjectCo-
>put_Value(bstrTemp, vTmp);
                    if (!SUCCEEDED(hr)) goto Error;

                    bstrTemp =
"JustInTimeActivation";
                    bTmp = TRUE;
                    vTmp = bTmp;
                    hr = pCatalogObjectCo-
>put_Value(bstrTemp, vTmp);
                    if (!SUCCEEDED(hr)) goto Error;

                    bstrTemp = "MaxPoolSize";
                    vTmp.Clear();       // clear
variant so it isn't stored as a bool (_variant_t
feature)
                    vTmp = (long)30;
                    hr = pCatalogObjectCo-
>put_Value(bstrTemp, vTmp);
                    if (!SUCCEEDED(hr)) goto Error;

                    bstrTemp =
"ObjectPoolingEnabled";
                    bTmp = TRUE;
                    vTmp = bTmp;
                    hr = pCatalogObjectCo-
>put_Value(bstrTemp, vTmp);
                    if (!SUCCEEDED(hr)) goto Error;



                    // save key to get the
InterfacesForComponent collection
                    hr = pCatalogObjectCo-
>get_Key(&vKey);
                    if (!SUCCEEDED(hr)) goto Error;

                    bstrTemp =
"InterfacesForComponent";
                    hr = pCatalogCollectionCo-
>GetCollection(bstrTemp, vKey, (IDispatch**)
&pCatalogCollectionItf);
                    if (!SUCCEEDED(hr)) goto Error;

                    hr = pCatalogCollectionItf-
>Populate();
                    if (!SUCCEEDED(hr)) goto Error;

                    hr = pCatalogCollectionItf-
>get_Count(&lCountItf);
                    if (!SUCCEEDED(hr)) goto Error;

                    // iterate through interfaces in
component
                    while (lCountItf > 0)
                    {
                              hr =
pCatalogCollectionItf->get_Item(lCountItf - 1,
(IDispatch**) &pCatalogObjectItf);
                              if (!SUCCEEDED(hr))
goto Error;

                              // save key to get the
MethodsForInterface collection
                              hr = pCatalogObjectItf-
>get_Key(&vKey);
```

```
                                if (!SUCCEEDED(hr))
goto Error;

                                bstrTemp =
"MethodsForInterface";
                                hr =
pCatalogCollectionItf->GetCollection(bstrTemp, vKey,
(IDispatch**) &pCatalogCollectionMethod);
                                if (!SUCCEEDED(hr))
goto Error;

                                hr =
pCatalogCollectionMethod->Populate();
                                if (!SUCCEEDED(hr))
goto Error;

                                hr =
pCatalogCollectionMethod->get_Count(&lCountMethod);
                                if (!SUCCEEDED(hr))
goto Error;

                                // iterate through
methods of interface
                                while (lCountMethod >
0)
                                {
                                        hr =
pCatalogCollectionMethod->get_Item(lCountMethod - 1,
(IDispatch**) &pCatalogObjectMethod);
                                        if
(!SUCCEEDED(hr)) goto Error;

                                        bstrTemp =
"AutoComplete";
                                        bTmp = TRUE;
                                        vTmp = bTmp;
                                        hr =
pCatalogObjectMethod->put_Value(bstrTemp, vTmp);
                                        if
(!SUCCEEDED(hr)) goto Error;


                pCatalogObjectMethod->Release();

                pCatalogObjectMethod = NULL;

                                        lCountMethod-
-;
                                }

                                // save changes
                                hr =
pCatalogCollectionMethod->SaveChanges(&lActProp);
                                if (!SUCCEEDED(hr))
goto Error;

                                pCatalogObjectItf-
>Release();
                                pCatalogObjectItf =
NULL;

                                lCountItf--;
                        }


                        pCatalogObjectCo->Release();
                        pCatalogObjectCo = NULL;

                        lCountCo--;
                }


                // save changes
                hr = pCatalogCollectionCo-
>SaveChanges(&lActProp);
                if (!SUCCEEDED(hr)) goto Error;

                pCatalogCollectionApp->Release();
                pCatalogCollectionApp = NULL;

                pCatalogCollectionCo->Release();
                pCatalogCollectionCo = NULL;

                pCatalogCollectionItf->Release();
                pCatalogCollectionItf = NULL;

                pCatalogCollectionMethod->Release();
                pCatalogCollectionMethod = NULL;


Error:
                CoUninitialize();

                if (!SUCCEEDED(hr))
                {
                        LPTSTR lpBuf;
                        DWORD dwRes =
FormatMessage(FORMAT_MESSAGE_ALLOCATE_BUFFER |
FORMAT_MESSAGE_FROM_SYSTEM,

                                                NULL,

                                                hr,


                MAKELANGID(LANG_NEUTRAL, SUBLANG_DEFAULT),

                                                (LPTSTR)
&lpBuf,

                                                0,

                                                NULL);
//                      _tprintf(__T("Error adding
components.  HRESULT: 0x%x\n%s"), hr, lpBuf);
                        return TRUE;
                }
                else
                        return FALSE;
}
```

Methods.h

```
/*        FILE:                 METHODS.H
 *                                      Microsoft
TPC-C Kit Ver. 4.69.000
 *                                      Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                                      not yet
audited
 *
 *        PURPOSE:  Header file for COM components.
 *
 *  Change history:
 *                  4.20.000 - first version
 *                  4.69.000 - updated rev number to
match kit
 */


enum COMPONENT_ERROR
{
          ERR_MISSING_REGISTRY_ENTRIES = 1,
          ERR_LOADDLL_FAILED,
          ERR_GETPROCADDR_FAILED,
          ERR_UNKNOWN_DB_PROTOCOL,
          ERR_MEM_ALLOC_FAILED
};

class CCOMPONENT_ERR : public CBaseErr
{
          public:
                    CCOMPONENT_ERR(COMPONENT_ERROR
Err)
                    {
                              m_Error = Err;
                              m_szTextDetail = NULL;
                              m_SystemErr = 0;
                              m_szErrorText = NULL;
                    };

                    CCOMPONENT_ERR(COMPONENT_ERROR
Err, char *szTextDetail, DWORD dwSystemErr)
                    {
                              m_Error = Err;
                              m_szTextDetail = new
char[strlen(szTextDetail)+1];
                              strcpy( m_szTextDetail,
szTextDetail );
                              m_SystemErr =
dwSystemErr;
                              m_szErrorText = NULL;
                    };

                    ~CCOMPONENT_ERR()
                    {
                              if (m_szTextDetail !=
NULL)
                                        delete []
m_szTextDetail;
```

```
                              if (m_szErrorText !=
NULL)
                                        delete []
m_szErrorText;
                    };

                    COMPONENT_ERROR     m_Error;
                    char
          *m_szTextDetail;
                    char
          *m_szErrorText;
                    DWORD
          m_SystemErr;

                    int ErrorType() {return
ERR_TYPE_COMPONENT;};
                    char *ErrorTypeStr() { return
"COMPONENT"; }
                    int ErrorNum() {return m_Error;};
                    char *ErrorText();
};


static void WriteMessageToEventLog(LPTSTR lpszMsg);


/////////////////////////////////////////////////
//////////////////////
// CTPCC_Common
class CTPCC_Common :
          public ITPCC,
          public IObjectControl,
          public IObjectConstruct,
          public
CComObjectRootEx<CComSingleThreadModel>
{
public:
BEGIN_COM_MAP(CTPCC_Common)
          COM_INTERFACE_ENTRY(ITPCC)
          COM_INTERFACE_ENTRY(IObjectControl)
          COM_INTERFACE_ENTRY(IObjectConstruct)
END_COM_MAP()

          CTPCC_Common();
          ~CTPCC_Common();

// ITPCC
public:
          HRESULT __stdcall NewOrder(
          VARIANT txn_in, VARIANT* txn_out);
          HRESULT __stdcall Payment(
          VARIANT txn_in, VARIANT* txn_out);
          HRESULT __stdcall Delivery(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
          HRESULT __stdcall StockLevel( VARIANT
txn_in, VARIANT* txn_out);
          HRESULT __stdcall OrderStatus(
          VARIANT txn_in, VARIANT* txn_out);

          HRESULT __stdcall CallSetComplete();

// IObjectControl
          STDMETHODIMP_(BOOL) CanBePooled() { return
m_bCanBePooled; }
          STDMETHODIMP Activate() { return S_OK; }
          // we don't support COM Services
transactions (no enlistment)
          STDMETHODIMP_(void) Deactivate() { /*
nothing to do */ }

// IObjectConstruct
          STDMETHODIMP Construct(IDispatch * pUnk);

          // helper methods
private:
          BOOL                m_bCanBePooled;
          CTPCC_BASE          *m_pTxn;

          struct COM_DATA
          {
                    int retval;
                    int error;
                    union
                    {
                              NEW_ORDER_DATA
          NewOrder;
                              PAYMENT_DATA
          Payment;
                              DELIVERY_DATA
          Delivery;
                              STOCK_LEVEL_DATA
          StockLevel;
                              ORDER_STATUS_DATA
          OrderStatus;
                    } u;
          };

};

/////////////////////////////////////////////////
//////////////////////
// CTPCC
class CTPCC :
          public CTPCC_Common,
          public CComCoClass<CTPCC, &CLSID_TPCC>
{
public:
DECLARE_REGISTRY_RESOURCEID(IDR_TPCC)

BEGIN_COM_MAP(CTPCC)
          //COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx<CComSingleThreadModel>)
          COM_INTERFACE_ENTRY2(IUnknown, ITPCC)
          COM_INTERFACE_ENTRY_CHAIN(CTPCC_Common)
END_COM_MAP()

};


/////////////////////////////////////////////////
//////////////////////
// CNewOrder
class CNewOrder :
          public CTPCC_Common,
          public CComCoClass<CNewOrder,
&CLSID_NewOrder>
{
public:
DECLARE_REGISTRY_RESOURCEID(IDR_NEWORDER)

BEGIN_COM_MAP(CNewOrder)
//        COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
          COM_INTERFACE_ENTRY2(IUnknown, ITPCC)
          COM_INTERFACE_ENTRY_CHAIN(CTPCC_Common)
END_COM_MAP()

// ITPCC
public:
//        HRESULT __stdcall NewOrder(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
          HRESULT __stdcall Payment(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
          HRESULT __stdcall StockLevel( VARIANT
txn_in, VARIANT* txn_out) {return E_NOTIMPL;}
          HRESULT __stdcall OrderStatus(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
};


          ////////////////////////////////////////
//////////////////////////////////
// COrderStatus
class COrderStatus :
          public CTPCC_Common,
          public CComCoClass<COrderStatus,
&CLSID_OrderStatus>
{
public:
DECLARE_REGISTRY_RESOURCEID(IDR_ORDERSTATUS)

BEGIN_COM_MAP(COrderStatus)
//        COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
          COM_INTERFACE_ENTRY2(IUnknown, ITPCC)
          COM_INTERFACE_ENTRY_CHAIN(CTPCC_Common)
END_COM_MAP()

// ITPCC
public:
          HRESULT __stdcall NewOrder(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
          HRESULT __stdcall Payment(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
          HRESULT __stdcall StockLevel( VARIANT
txn_in, VARIANT* txn_out) {return E_NOTIMPL;}
//        HRESULT __stdcall OrderStatus(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
};
```

```
/////////////////////////////////////////////////
//////////////////////
// CPayment
class CPayment :
          public CTPCC_Common,
          public CComCoClass<CPayment,
&CLSID_Payment>
{
public:
DECLARE_REGISTRY_RESOURCEID(IDR_PAYMENT)

BEGIN_COM_MAP(CPayment)
//        COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
          COM_INTERFACE_ENTRY2(IUnknown, ITPCC)
          COM_INTERFACE_ENTRY_CHAIN(CTPCC_Common)
END_COM_MAP()

// ITPCC
public:
          HRESULT __stdcall NewOrder(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
//        HRESULT __stdcall Payment(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
          HRESULT __stdcall StockLevel( VARIANT
txn_in, VARIANT* txn_out) {return E_NOTIMPL;}
          HRESULT __stdcall OrderStatus(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
};


/////////////////////////////////////////////////
//////////////////////
// CStockLevel
class CStockLevel :
          public CTPCC_Common,
          public CComCoClass<CStockLevel,
&CLSID_StockLevel>
{
public:
DECLARE_REGISTRY_RESOURCEID(IDR_STOCKLEVEL)

BEGIN_COM_MAP(CStockLevel)
//        COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
          COM_INTERFACE_ENTRY2(IUnknown, ITPCC)
          COM_INTERFACE_ENTRY_CHAIN(CTPCC_Common)
END_COM_MAP()

// ITPCC
public:
          HRESULT __stdcall NewOrder(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
          HRESULT __stdcall Payment(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
//        HRESULT __stdcall StockLevel( VARIANT
txn_in, VARIANT* txn_out) {return E_NOTIMPL;}
          HRESULT __stdcall OrderStatus(
          VARIANT txn_in, VARIANT* txn_out) {return
E_NOTIMPL;}
};
```

RCa03544

```
#line
1"C:\\temp\\MSTPCC.442\\WEBCLNT\\install\\src\\instal
l.rc"
#line 1
//Microsoft Developer Studio generated resource
script.
//
#include "resource.h"
#line 5
#define APSTUDIO_READONLY_SYMBOLS
/////////////////////////////////////////////////
//////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"
#line 12
/////////////////////////////////////////////////
//////////////////////
#undef APSTUDIO_READONLY_SYMBOLS
#line 15
/////////////////////////////////////////////////
//////////////////////
// English (U.S.) resources
#line 18
#if !defined(AFX_RESOURCE_DLL) ||
defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32
#line 24
/////////////////////////////////////////////////
//////////////////////
//
// Dialog
//
#line 29
IDD_DIALOG1 DIALOGEX 0, 0, 219, 351
STYLE DS_MODALFRAME | DS_CENTER | WS_MINIMIZEBOX |
WS_POPUP | WS_CAPTION |
WS_SYSMENU
CAPTION "TPC-C Web Client Installation Utility"
FONT 8, "MS Sans Serif"
BEGIN
EDITTEXT        ED_THREADS,164,45,34,12,ES_RIGHT |
ES_NUMBER,
WS_EX_RTLREADING
EDITTEXT
ED_MAXDELIVERIES,164,59,34,12,ES_RIGHT | ES_NUMBER,
WS_EX_RTLREADING
EDITTEXT
ED_MAXCONNECTION,164,73,34,12,ES_RIGHT | ES_NUMBER,
WS_EX_RTLREADING
CONTROL
"None",IDC_TM_NONE,"Button",BS_AUTORADIOBUTTON |
WS_GROUP | WS_TABSTOP,43,100,33,10
CONTROL
"COM",IDC_TM_MTS,"Button",BS_AUTORADIOBUTTON |
WS_TABSTOP,43,113,32,10
CONTROL
"TUXEDO",IDC_TM_TUXEDO,"Button",BS_AUTORADIOBUTTON |
WS_TABSTOP,106,100,46,10
CONTROL
"ENCINA",IDC_TM_ENCINA,"Button",BS_AUTORADIOBUTTON |
WS_DISABLED | WS_TABSTOP,106,113,43,10
EDITTEXT
ED_DB_SERVER,131,152,67,12,ES_AUTOHSCROLL
EDITTEXT
ED_DB_USER_ID,131,165,67,12,ES_AUTOHSCROLL
EDITTEXT
ED_DB_PASSWORD,131,178,67,12,ES_AUTOHSCROLL
EDITTEXT
ED_DB_NAME,131,191,67,12,ES_AUTOHSCROLL
CONTROL
"DBLIB",IDC_DBLIB,"Button",BS_AUTORADIOBUTTON |
WS_GROUP |
WS_TABSTOP,45,219,39,12
CONTROL
"ODBC",IDC_ODBC,"Button",BS_AUTORADIOBUTTON |
WS_TABSTOP,
91,219,39,12
EDITTEXT
ED_IIS_MAX_THEAD_POOL_LIMIT,164,263,34,12,ES_RIGHT |
ES_NUMBER,WS_EX_RTLREADING
EDITTEXT
ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE,164,277,34,12,ES_RI
GHT |
ES_NUMBER,WS_EX_RTLREADING
EDITTEXT
ED_IIS_THREAD_TIMEOUT,164,291,34,12,ES_RIGHT |
ES_NUMBER,
WS_EX_RTLREADING
EDITTEXT
ED_IIS_LISTEN_BACKLOG,164,305,34,12,ES_RIGHT |
ES_NUMBER,
WS_EX_RTLREADING
DEFPUSHBUTTON   "OK",IDOK,53,331,50,14
PUSHBUTTON      "Cancel",IDCANCEL,119,331,50,14
EDITTEXT        IDC_PATH,106,26,91,13,ES_AUTOHSCROLL
| ES_READONLY
LTEXT           "Number of Delivery
Threads:",IDC_STATIC,35,45,115,12
LTEXT           "Max Number of
Connections:",IDC_STATIC,35,73,115,12
RTEXT           "Version 4.11",IDC_VERSION,120,4,89,9
LTEXT           "IIS Max Thread Pool
Limit:",IDC_STATIC,36,263,115,12
LTEXT           "Web Service Backlog Queue
Size:",IDC_STATIC,36,277,115,
12
LTEXT           "IIS Thread Timeout
(seconds):",IDC_STATIC,36,291,115,12
LTEXT           "IIS Listen
Backlog:",IDC_STATIC,36,307,115,10
```

```
GROUPBOX        "Database
Interface",IDC_STATIC,35,208,163,27,WS_GROUP
LTEXT           "Installation
directory:",IDC_STATIC,35,29,71,10
GROUPBOX        "Transaction
Monitor",IDC_STATIC,33,90,165,37
LTEXT           "Server Name:",IDC_STATIC,35,155,56,8
LTEXT           "User ID:",IDC_STATIC,35,168,60,8
LTEXT           "User
Password:",IDC_STATIC,35,181,83,8
LTEXT           "Database
Name:",IDC_STATIC,35,194,54,8
GROUPBOX        "SQL Server Connection
Properties",IDC_STATIC,22,139,187,
102
GROUPBOX        "Web Client
Properties",IDC_STATIC,22,15,187,118
GROUPBOX        "IIS
Settings",IDC_STATIC,22,247,187,79
LTEXT           "Max Pending
Deliveries:",IDC_STATIC,35,59,115,12
END
#line 90
IDD_DIALOG2 DIALOGEX 0, 0, 117, 62
STYLE DS_SETFOREGROUND | DS_3DLOOK | DS_CENTER |
WS_POPUP | WS_BORDER
EXSTYLE WS_EX_STATICEDGE
FONT 12, "MS Sans Serif", 0, 0, 0x1
BEGIN
DEFPUSHBUTTON   "OK",IDOK,33,45,50,9
CTEXT           "HTML TPC-C Installation
Successfull",IDC_RESULTS,7,22,
102,18,0,WS_EX_CLIENTEDGE
ICON
IDI_ICON2,IDC_STATIC,50,7,18,20,SS_REALSIZEIMAGE,
WS_EX_TRANSPARENT
END
#line 102
IDD_DIALOG3 DIALOG DISCARDABLE  0, 0, 91, 40
STYLE DS_SYSMODAL | DS_MODALFRAME | DS_3DLOOK |
DS_CENTER | WS_CAPTION
CAPTION "Installing TPC-C Web Client"
FONT 12, "Arial Black"
BEGIN
CONTROL
"Progress1",IDC_PROGRESS1,"msctls_progress32",WS_BORD
ER,
7,20,77,13
CTEXT
"Static",IDC_STATUS,7,7,77,12,SS_SUNKEN
END
#line 112
IDD_DIALOG4 DIALOG DISCARDABLE  0, 0, 291, 202
STYLE DS_MODALFRAME | DS_CENTER | WS_POPUP |
WS_CAPTION | WS_SYSMENU
CAPTION "Client End User License"
FONT 8, "MS Sans Serif"
BEGIN
EDITTEXT        IDC_LICENSE,7,7,271,167,ES_MULTILINE
| ES_AUTOVSCROLL |
ES_AUTOHSCROLL | ES_READONLY | WS_VSCROLL |
WS_HSCROLL
DEFPUSHBUTTON   "I &Agree",IDOK,87,181,50,14
PUSHBUTTON      "&Cancel",IDCANCEL,153,181,50,14
END
#line 124
//////////////////////////////////////////////////
//////////////////////
//
// DESIGNINFO
//
#line 129
#ifdef APSTUDIO_INVOKED
GUIDELINES DESIGNINFO DISCARDABLE
BEGIN
IDD_DIALOG1, DIALOG
BEGIN
LEFTMARGIN, 22
RIGHTMARGIN, 209
VERTGUIDE, 35
VERTGUIDE, 198
TOPMARGIN, 4
BOTTOMMARGIN, 345
END
#line 142
IDD_DIALOG2, DIALOG
BEGIN
LEFTMARGIN, 7
RIGHTMARGIN, 109
TOPMARGIN, 7
BOTTOMMARGIN, 54
END
#line 150
IDD_DIALOG3, DIALOG
BEGIN
LEFTMARGIN, 7
RIGHTMARGIN, 84
TOPMARGIN, 7
BOTTOMMARGIN, 33
END
#line 158
IDD_DIALOG4, DIALOG
BEGIN
LEFTMARGIN, 7
RIGHTMARGIN, 278
TOPMARGIN, 7
BOTTOMMARGIN, 195
END
END
#endif    // APSTUDIO_INVOKED
#line 169
#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////
//////////////////////
//
// TEXTINCLUDE
//
#line 175
1 TEXTINCLUDE DISCARDABLE
BEGIN
"resource.h\0"
END
#line 180
2 TEXTINCLUDE DISCARDABLE
BEGIN
"#include ""afxres.h""\r\n"
"\0"
END
#line 186
3 TEXTINCLUDE DISCARDABLE
BEGIN
"\r\n"
"\0"
END
#line 192
#endif    // APSTUDIO_INVOKED
#line 195
//////////////////////////////////////////////////
//////////////////////
//
// Icon
//
#line 200
// Icon with lowest ID value placed first to ensure
application icon
// remains consistent on all systems.
IDI_ICON1               ICON    DISCARDABLE
"icon1.ico"
IDI_ICON2               ICON    DISCARDABLE
"icon2.ico"
#line 205
//////////////////////////////////////////////////
//////////////////////
//
// TPCCDLL
//
#line 210
IDR_TPCCDLL             TPCCDLL DISCARDABLE
"..\\..\\isapi_dll\\bin\\tpcc.dll"
#line 212
#ifndef _MAC
//////////////////////////////////////////////////
//////////////////////
//
// Version
//
#line 218
VS_VERSION_INFO VERSIONINFO
FILEVERSION 0,4,20,0
PRODUCTVERSION 0,4,20,0
FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
FILEFLAGS 0x1L
#else
FILEFLAGS 0x0L
#endif
FILEOS 0x40004L
FILETYPE 0x1L
FILESUBTYPE 0x0L
BEGIN
BLOCK "StringFileInfo"
BEGIN
BLOCK "040904b0"
BEGIN
VALUE "Comments", "TPC-C Web Client Installer\0"
VALUE "CompanyName", "Microsoft\0"
VALUE "FileDescription", "install\0"
VALUE "FileVersion", "0, 4, 20, 0\0"
VALUE "InternalName", "install\0"
```

```
VALUE "LegalCopyright", "Copyright ¬ 1999\0"
VALUE "OriginalFilename", "install.exe\0"
VALUE "ProductName", "Microsoft install\0"
VALUE "ProductVersion", "0, 4, 20, 0\0"
END
END
BLOCK "VarFileInfo"
BEGIN
VALUE "Translation", 0x409, 1200
END
END
#line 252
#endif    // !_MAC
#line 255
////////////////////////////////////////////////
//////////////////////
//
// LICENSE
//
#line 260
IDR_LICENSE1            LICENSE DISCARDABLE
"license.txt"
#line 262
////////////////////////////////////////////////
//////////////////////
//
// DBLIB_DLL
//
#line 267
IDR_DBLIB_DLL           DBLIB_DLL DISCARDABLE
"..\\..\\db_dblib_dll\\bin\\tpcc_dblib.dll"
#line 269
////////////////////////////////////////////////
//////////////////////
//
// ODBC_DLL
//
#line 274
IDR_ODBC_DLL            ODBC_DLL DISCARDABLE
"..\\..\\db_odbc_dll\\bin\\tpcc_odbc.dll"
#line 276
////////////////////////////////////////////////
//////////////////////
//
// TUXEDO_APP
//
#line 281
IDR_TUXEDO_APP          TUXEDO_APP DISCARDABLE
"..\\..\\tuxapp\\bin\\tuxapp.exe"
#line 283
////////////////////////////////////////////////
//////////////////////
//
// TUXEDO_DLL
//
#line 288
IDR_TUXEDO_DLL          TUXEDO_DLL DISCARDABLE
"..\\..\\tm_tuxedo_dll\\bin\\tpcc_tuxedo.dll"
#line 290
////////////////////////////////////////////////
//////////////////////
//
// COM_DLL
//
#line 295
IDR_COM_DLL             COM_DLL DISCARDABLE
"..\\..\\tm_com_dll\\bin\\tpcc_com.dll"
#line 297
////////////////////////////////////////////////
//////////////////////
//
// COM_PS_DLL
//
#line 302
IDR_COMPS_DLL           COM_PS_DLL DISCARDABLE
"..\\..\\tpcc_com_ps\\bin\\tpcc_com_ps.dll"
#line 304
////////////////////////////////////////////////
//////////////////////
//
// COM_ALL_DLL
//
#line 309
IDR_COMALL_DLL          COM_ALL_DLL DISCARDABLE
"..\\..\\tpcc_com_all\\bin\\tpcc_com_all.dll"
#line 311
////////////////////////////////////////////////
//////////////////////
//
// COM_TYPLIB
//
#line 316
IDR_COMTYPLIB_DLL       COM_TYPLIB DISCARDABLE
"..\\..\\tpcc_com_all\\src\\tpcc_com_all.tlb"
#line 318
#endif    // English (U.S.) resources
////////////////////////////////////////////////
//////////////////////
#line 323
#ifndef APSTUDIO_INVOKED
////////////////////////////////////////////////
//////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
#line 330
////////////////////////////////////////////////
//////////////////////
#endif    // not APSTUDIO_INVOKED
```

ReadRegistry.cpp

```
/*        FILE:               READREGISTRY.CPP
 *                                    Microsoft
TPC-C Kit Ver. 4.20.000
 *                                    Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                                    not yet
audited
 *
 *        PURPOSE:  Implementation for TPC-C class.
 *        Contact:  Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *                  4.20.000 - first version
 */


/* FUNCTION: ReadTPCCRegistrySettings
 *
 * PURPOSE:         This function reads the NT
registry for startup parameters. There parameters are
 *                           under the TPCC key.
 *
 * RETURNS          FALSE = no errors
 *                           TRUE  = error reading
registry
 */
BOOL ReadTPCCRegistrySettings( TPCCREGISTRYDATA *pReg
)
{
          HKEY      hKey;
          DWORD     size;
          DWORD     type;
          DWORD     dwTmp;
          char      szTmp[256];

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

          // determine database protocol to use;
always has to be ODBC
          pReg->eDB_Protocol = ODBC;
          size = sizeof(szTmp);
          //if ( RegQueryValueEx(hKey, "DB_Protocol",
0, &type, (BYTE *)&szTmp, &size) == ERROR_SUCCESS )
          //{
                    //if ( !stricmp(szTmp,
szDBNames[ODBC]) )
                    //        pReg->eDB_Protocol =
ODBC;
          //}

          pReg->eTxnMon = None;
          // determine txn monitor to use; may be
either COM, or blank
          size = sizeof(szTmp);
          if ( RegQueryValueEx(hKey, "TxnMonitor", 0,
&type, (BYTE *)&szTmp, &size) == ERROR_SUCCESS )
          {
                    if ( !stricmp(szTmp,
szTxnMonNames[COM]) )
                              pReg->eTxnMon = COM;
          }

          pReg->bCOM_SinglePool = FALSE;
          size = sizeof(szTmp);
          if ( RegQueryValueEx(hKey,
"COM_SinglePool", 0, &type, (BYTE *)&szTmp, &size) ==
ERROR_SUCCESS )
```

```
        {
                if ( !stricmp(szTmp, "YES") )
                        pReg->bCOM_SinglePool =
TRUE;
        }

        pReg->dwMaxConnections = 0;
        size = sizeof(dwTmp);
        if ( ( RegQueryValueEx(hKey,
"MaxConnections", 0, &type, (LPBYTE)&dwTmp, &size) ==
ERROR_SUCCESS )
                && (type == REG_DWORD) )
                pReg->dwMaxConnections = dwTmp;

        pReg->dwMaxPendingDeliveries = 0;
        size = sizeof(dwTmp);
        if ( ( RegQueryValueEx(hKey,
"MaxPendingDeliveries", 0, &type, (LPBYTE)&dwTmp,
&size) == ERROR_SUCCESS )
                && (type == REG_DWORD) )
                pReg->dwMaxPendingDeliveries =
dwTmp;

        pReg->dwNumberOfDeliveryThreads = 0;
        size = sizeof(dwTmp);
        if ( ( RegQueryValueEx(hKey,
"NumberOfDeliveryThreads", 0, &type, (LPBYTE)&dwTmp,
&size) == ERROR_SUCCESS )
                && (type == REG_DWORD) )
                pReg->dwNumberOfDeliveryThreads =
dwTmp;

        size = sizeof( pReg->szPath );
        if ( RegQueryValueEx(hKey, "Path", 0,
&type, (BYTE *)&pReg->szPath, &size) != ERROR_SUCCESS
)
                pReg->szPath[0] = 0;

        size = sizeof( pReg->szDbServer );
        if ( RegQueryValueEx(hKey, "DbServer", 0,
&type, (BYTE *)&pReg->szDbServer, &size) !=
ERROR_SUCCESS )
                pReg->szDbServer[0] = 0;

        size = sizeof( pReg->szDbName );
        if ( RegQueryValueEx(hKey, "DbName", 0,
&type, (BYTE *)&pReg->szDbName, &size) !=
ERROR_SUCCESS )
                pReg->szDbName[0] = 0;

        size = sizeof( pReg->szDbUser );
        if ( RegQueryValueEx(hKey, "DbUser", 0,
&type, (BYTE *)&pReg->szDbUser, &size) !=
ERROR_SUCCESS )
                pReg->szDbUser[0] = 0;

        size = sizeof( pReg->szDbPassword );
        if ( RegQueryValueEx(hKey, "DbPassword", 0,
&type, (BYTE *)&pReg->szDbPassword, &size) !=
ERROR_SUCCESS )
                pReg->szDbPassword[0] = 0;

        size = sizeof( pReg->szSPPrefix );
        if ( RegQueryValueExW(hKey, L"SPPrefix", 0,
&type, (BYTE *)&pReg->szSPPrefix, &size) !=
ERROR_SUCCESS )
                pReg->szSPPrefix[0] = L'\0';

        pReg->dwConnectDelay = 0;
        size = sizeof(dwTmp);
        if ( ( RegQueryValueEx(hKey,
"ConnectDelay", 0, &type, (LPBYTE)&dwTmp, &size) ==
ERROR_SUCCESS )
                && (type == REG_DWORD) )
                pReg->dwConnectDelay = dwTmp;

        pReg->bCallNoDuplicatesNewOrder = FALSE;
        size = sizeof(dwTmp);
        if ( ( RegQueryValueEx(hKey,
"CallNoDuplicatesNewOrder", 0, &type, (LPBYTE)&dwTmp,
&size) == ERROR_SUCCESS )
                && (type == REG_DWORD) )
                pReg->bCallNoDuplicatesNewOrder =
dwTmp;

        RegCloseKey(hKey);

        return FALSE;
}
```

ReadRegistry.h

```
/*      FILE:               ReadRegistry.h
 *                                  Microsoft
TPC-C Kit Ver. 4.69.000
 *                                  Copyright
Microsoft, 1999
 *            All Rights Reserved
 *
 *                                  not audited
 *
 *      PURPOSE:  Header for registry related code.
 *
 *  Change history:
 *                4.20.000 - first version
 *                4.69.000 - updated rev number to
match kit
 */

enum DBPROTOCOL { Unspecified, ODBC };
const char *szDBNames[] = { "Unspecified", "ODBC" };

enum TXNMON { None, COM };
const char *szTxnMonNames[] = { "NONE", "COM" };

//This structure defines the data necessary to keep
distinct for each terminal or client connection.
typedef struct _TPCCREGISTRYDATA
{
        enum DBPROTOCOL eDB_Protocol;
        enum TXNMON eTxnMon;
        BOOL bCOM_SinglePool;
        DWORD dwMaxConnections;
        DWORD dwMaxPendingDeliveries;
        DWORD dwNumberOfDeliveryThreads;
        char szPath[128];
        char szDbServer[32];
        char szDbName[32];
        char szDbUser[32];
        char szDbPassword[32];
        wchar_t szSPPrefix[32];
        //tpcc_odbc.dll stored procedures prefix
        DWORD dwConnectDelay;         // delay in
ms to use in pacing connection open and close
        BOOL bCallNoDuplicatesNewOrder;         //
whether to check for non-duplicate item ids and call
a different New Order SP
} TPCCREGISTRYDATA, *PTPCCREGISTRYDATA;

BOOL ReadTPCCRegistrySettings( TPCCREGISTRYDATA *pReg
);
```

resource (2).h

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by tpcc.rc
//
#define IDD_DIALOG1                     101

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        102
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1000
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

resource (3).h

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by tpcc_com_all.rc
//
#define IDS_PROJNAME                    100
#define IDR_TPCC                        101
#define IDR_NEWORDER                    102
#define IDR_ORDERSTATUS                 103
#define IDR_PAYMENT                     104
#define IDR_STOCKLEVEL                  105

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        202
#define _APS_NEXT_COMMAND_VALUE         32768
#define _APS_NEXT_CONTROL_VALUE         201
```

```
#define _APS_NEXT_SYMED_VALUE           106
#endif
#endif
```

RESOURCE.H

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by install.rc
//
#define IDD_DIALOG1                     101
#define IDI_ICON1                       102
#define IDR_TPCCDLL                     103
#define IDD_DIALOG2                     105
#define IDI_ICON2                       106
#define IDR_DELIVERY                    107
#define IDD_DIALOG3                     108
#define IDR_LICENSE1                    112
#define IDD_DIALOG4                     113
#define IDR_TPCCOBJ1                    117
#define IDR_TPCCSTUB1                   118
#define IDR_ODBC_DLL                    123
#define IDR_COM_DLL                     126
#define IDR_COMPS_DLL                   127
#define IDR_COMALL_DLL                  128
#define IDR_COMTYPLIB_DLL               129
#define IDR_MSVCR71                     130
#define BN_LOG                          1001
#define ED_KEEP                         1002
#define ED_THREADS                      1003
#define ED_THREADS2                     1004
#define IDC_PATH                        1007
#define IDC_VERSION                     1009
#define IDC_RESULTS                     1010
#define IDC_PROGRESS1                   1011
#define IDC_STATUS                      1012
#define IDC_BUTTON1                     1013
#define ED_MAXCONNECTION                1014
#define ED_IIS_MAX_THEAD_POOL_LIMIT     1015
#define ED_MAXDELIVERIES                1016
#define ED_WEB_SERVICE_BACKLOG_QUEUE_SIZE 1017
#define ED_IIS_THREAD_TIMEOUT           1018
#define ED_IIS_LISTEN_BACKLOG           1019
#define IDC_DBLIB                       1021
#define IDC_LICENSE                     1022
#define IDC_ODBC                        1022
#define IDC_CONNECT_POOL                1023
#define ED_DB_SERVER                    1023
#define ED_USER_CONNECT_DELAY_TIME      1024
#define ED_DB_USER_ID                   1024
#define IDC_MTS                         1025
#define IDC_TM_MTS                      1025
#define IDC_TM_TUXEDO                   1026
#define IDC_TM_NONE                     1027
#define ED_DB_PASSWORD                  1028
#define ED_DB_NAME                      1029
#define IDC_TM_ENCINA                   1030

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        131
#define _APS_NEXT_COMMAND_VALUE         40001
#define _APS_NEXT_CONTROL_VALUE         1031
#define _APS_NEXT_SYMED_VALUE           101
#endif
#endif
```

tpcc.cpp

```
/*        FILE:               TPCC.C
 *                                      Microsoft
TPC-C Kit Ver. 4.69.000
 *                                      Copyright
Microsoft, 1999
 *             All Rights Reserved
 *
 *                                      Version
4.10.000 audited by Richard Gimarc, Performance
Metrics, 3/17/99
 *
 *        PURPOSE:  Main module for TPCC.DLL which is
an ISAPI service dll.
 *        Contact:  Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *                  4.20.000 - reworked error
handling; added options for COM and Encina txn
monitors
 *                  4.69.000 - updated rev number to
match kit
 */

#include <windows.h>
#include <process.h>
#include <tchar.h>
#include <stdio.h>
#include <stdarg.h>
#include <malloc.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include <io.h>
#include <assert.h>

#include <sqltypes.h>

#ifdef ICECAP
#include <icapexp.h>
#endif

#include "..\..\common\src\trans.h"
          //tpckit transaction header contains
definations of structures specific to TPC-C
#include "..\..\common\src\error.h"
#include "..\..\common\src\txn_base.h"
#include "..\..\common\src\ReadRegistry.h"
#include "..\..\common\txnlog\include\rtetime.h"
#include "..\..\common\txnlog\include\spinlock.h"
#include "..\..\common\txnlog\include\txnlog.h"

// Database layer includes
#include "..\..\db_odbc_dll\src\tpcc_odbc.h"
          // ODBC implementation of TPC-C txns

// Txn monitor layer includes
#include "..\..\tm_com_dll\src\tpcc_com.h"
                    // COM Services implementation on
TPC-C txns

#include "httpext.h"
          //ISAPI DLL information header
#include "tpcc.h"
          //this dlls specific structure, value e.t.
header.


#define   LEN_ERR_STRING      256

// defines for Make<Txn>Form calls to distinguish
input and output flavors
#define   OUTPUT_FORM         0
#define INPUT_FORM  1

char
          szMyComputerName[MAX_COMPUTERNAME_LENGTH+1]
;

//Terminal client id structure
TERM      Term = { 0, 0, 0, NULL };

// The WEBCLIENT_VERSION string specifies the version
level of this web client interface.
// The RTE must be synchronized with the interface
level on login, otherwise the login
// will fail.  This is a sanity check to catch
problems resulting from mismatched versions
// of the RTE and web client.
#define WEBCLIENT_VERSION "420"

static    CRITICAL_SECTION
          TermCriticalSection;

static  HINSTANCE hLibInstanceTm = NULL;
static  HINSTANCE hLibInstanceDb = NULL;

TYPE_CTPCC_ODBC               *pCTPCC_ODBC_new;
TYPE_CTPCC_COM                *pCTPCC_COM_new;

// For deferred Delivery txns:

CTxnLog
          *txnDelilog = NULL;
          //used to log delivery transaction
information

HANDLE
          hWorkerSemaphore    = INVALID_HANDLE_VALUE;
```

```
HANDLE
        hDoneEvent                              =
INVALID_HANDLE_VALUE;
HANDLE
        *pDeliHandles                   = NULL;

// configuration settings from registry
TPCCREGISTRYDATA            Reg;


DWORD
        dwNumDeliveryThreads = 4;
CRITICAL_SECTION            DelBuffCriticalSection;
                //critical section for delivery
transactions cache
DELIVERY_TRANSACTION        *pDelBuff
        = NULL;
DWORD
        dwDelBuffSize               = 100;
        // size of circular buffer for delivery
txns
DWORD
        dwDelBuffFreeCount;
        // number of buffers free
DWORD
        dwDelBuffBusyIndex  = 0;                //
index position of entry waiting to be delivered
DWORD
        dwDelBuffFreeIndex  = 0;                //
index position of unused entry

// Critical section to synchronize connection open
and close.
//
CRITICAL_SECTION hConnectCriticalSection;

#include "..\..\common\src\ReadRegistry.cpp"

/* FUNCTION: DllMain
 *
 * PURPOSE:         This function is the entry point
for the DLL.  This implementation is based on the
 *                  fact that
DLL_PROCESS_ATTACH is only called from the inet
service once.
 *
 * ARGUMENTS:       HANDLE    hModule
                    module handle
 *                              DWORD
        ul_reason_for_call  reason for call
 *                              LPVOID
        lpReserved
        reserved for future use
 *
 * RETURNS:                   BOOL      FALSE
                            errors occured in
initialization
 *
        TRUE                                    DLL
successfully initialized
 */
```

```
BOOL APIENTRY DllMain(HANDLE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved)
{
        DWORD  i;
        char szEvent[LEN_ERR_STRING] = "\0";
        char szLogFile[128];
        char szDllName[128];

// debugging....
// DebugBreak();

        try
        {
                switch( ul_reason_for_call )
                {
                        case
DLL_PROCESS_ATTACH:
                                        {

        DWORD dwSize = MAX_COMPUTERNAME_LENGTH+1;

        GetComputerName(szMyComputerName, &dwSize);

        szMyComputerName[dwSize] = 0;
                                        }


        DisableThreadLibraryCalls((HMODULE)hModule)
;

        InitializeCriticalSection(&TermCriticalSect
ion);

                                        if (
ReadTPCCRegistrySettings( &Reg ) )

        throw new CWEBCLNT_ERR(
ERR_MISSING_REGISTRY_ENTRIES );

                                        dwDelBuffSize
= min( Reg.dwMaxPendingDeliveries, 10000 );  // min
with 10000 as a sanity constraint

        dwNumDeliveryThreads = min(
Reg.dwNumberOfDeliveryThreads, 100 );  // min with
100 as a sanity constraint

                                        TermInit();


                                        if
(Reg.eTxnMon == COM)
                                        {

        strcpy( szDllName, Reg.szPath );

        strcat( szDllName, "tpcc_com.dll");

        hLibInstanceTm = LoadLibrary( szDllName );
                                                if
(hLibInstanceTm == NULL)

        throw new CWEBCLNT_ERR( ERR_LOADDLL_FAILED,
szDllName, GetLastError() );
```

```
                                                //
get function pointer to wrapper for class constructor

        pCTPCC_COM_new = (TYPE_CTPCC_COM*)
GetProcAddress(hLibInstanceTm,"CTPCC_COM_new");
                                                if
(pCTPCC_COM_new == NULL)

        throw new CWEBCLNT_ERR(
ERR_GETPROCADDR_FAILED, szDllName, GetLastError() );
                                                }

                                                // load DLL
for database connection
                                                if
((Reg.eTxnMon == None) || (dwNumDeliveryThreads > 0))
                                                {
                                                        if
(Reg.eDB_Protocol == ODBC)
                                                        {

        strcpy( szDllName, Reg.szPath );

        strcat( szDllName, "tpcc_odbc.dll");

        hLibInstanceDb = LoadLibrary( szDllName );

        if (hLibInstanceDb == NULL)

                throw new CWEBCLNT_ERR(
ERR_LOADDLL_FAILED, szDllName, GetLastError() );


        // get function pointer to wrapper for
class constructor

        pCTPCC_ODBC_new = (TYPE_CTPCC_ODBC*)
GetProcAddress(hLibInstanceDb,"CTPCC_ODBC_new");

        if (pCTPCC_ODBC_new == NULL)

                throw new CWEBCLNT_ERR(
ERR_GETPROCADDR_FAILED, szDllName, GetLastError() );



                                                }
                                                // Check
whether Service Pack 1 has been installed if
                                                // running on
Windows Server 2003. The RTM version has
                                                // a
limitation on concurrent HTTP connections.
                                                //

        OSVERSIONINFOEX             VersionInfo;


        VersionInfo.dwOSVersionInfoSize =
sizeof(OSVERSIONINFOEX);
```

```
                                        if
(GetVersionEx((LPOSVERSIONINFO)&VersionInfo))
                                        {
                                                if
(VersionInfo.dwMajorVersion == 5 &&     // Windows
2000/2003 Server?

          VersionInfo.dwMinorVersion == 2 &&      //
Windows 2003 Server?

          VersionInfo.wServicePackMajor == 0)     //
Service Pack installed?
                                                  {

          TCHAR szMsg[256];


          _sntprintf(szMsg, sizeof(szMsg),

                                        "\nRunning on
Windows Server 2003 without at least Service Pack
1\n"

                                        "limits the
number of concurrent HTTP connections to around
8000");


          // Use event logging to log the error.

          //

          HANDLE    hEventSource =
RegisterEventSource(NULL, TEXT("TPCC.DLL"));

          LPTSTR  lpszStrings[1] = { szMsg };



          if (hEventSource != NULL)

          {

                    ReportEvent(hEventSource, //
handle of event source

                              EVENTLOG_WARNING_TYPE,
// event type

                              0,
// event category

                              0,
// event ID

                              NULL,
// current user's SID

                              1,
// strings in lpszStrings

                              0,
// no bytes of raw data

                              (LPCTSTR *)lpszStrings,
// array of error strings


                              NULL);
// no raw data


                    (VOID)
DeregisterEventSource(hEventSource);

          }

                                                  }
                                        }

                                        if
(dwNumDeliveryThreads)
                                        {
                                                //
Initialize delivery delay critical section
                                                //

          InitializeCriticalSection(&hConnectCritical
Section);

                                                //
for deferred delivery txns:

          hDoneEvent = CreateEvent( NULL, TRUE /*
manual reset */, FALSE /* initially not signalled */,
NULL );

          InitializeCriticalSection(&DelBuffCriticalS
ection);

          hWorkerSemaphore = CreateSemaphore( NULL,
0, dwDelBuffSize, NULL );

          dwDelBuffFreeCount = dwDelBuffSize;


          InitJulianTime(NULL);


                                                //
create unique log file name based on delilog-yymmdd-
hhmm.log

          SYSTEMTIME Time;

          GetLocalTime( &Time );

          wsprintf( szLogFile, "%sdelivery-
%2.2d%2.2d%2.2d-%2.2d%2.2d-%2.2ds%2.2dms.log",

          Reg.szPath, Time.wYear % 100, Time.wMonth,
Time.wDay, Time.wHour, Time.wMinute, Time.wSecond,
Time.wMilliseconds );

          txnDelilog = new CTxnLog(szLogFile,
TXN_LOG_WRITE);

          //write event into txn log for START

          txnDelilog-
>WriteCtrlRecToLog(TXN_EVENT_START, szMyComputerName,
sizeof(szMyComputerName));

                                                //
allocate structures for delivery buffers and thread
mgmt

          pDeliHandles = new
HANDLE[dwNumDeliveryThreads];

          pDelBuff = new
DELIVERY_TRANSACTION[dwDelBuffSize];
                                                //
launch DeliveryWorkerThread to perform actual
delivery txns

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

          pDeliHandles[i] = (HANDLE) _beginthread(
DeliveryWorkerThread, 0, NULL );

          if (pDeliHandles[i] ==
INVALID_HANDLE_VALUE)

                    throw new CWEBCLNT_ERR(
ERR_DELIVERY_THREAD_FAILED );
                                                  }
                                                }

                                                break;

                                        case
DLL_PROCESS_DETACH:
                                        if
(dwNumDeliveryThreads)
                                        {
                                                if
(txnDelilog != NULL)
                                                {

          //write event into txn log for STOP

          txnDelilog-
>WriteCtrlRecToLog(TXN_EVENT_STOP, szMyComputerName,
sizeof(szMyComputerName));


          // This will do a clean shutdown of the
delivery log file

          CTxnLog   *txnDelilogLocal = txnDelilog;

          txnDelilog= NULL;

          delete txnDelilogLocal;
                                                }
```

```
        delete [] pDeliHandles;

        delete [] pDelBuff;


        CloseHandle( hWorkerSemaphore );

        CloseHandle( hDoneEvent );

        DeleteCriticalSection(&DelBuffCriticalSecti
on);

                                                //
Delete delivery delay critical section
                                                //

        DeleteCriticalSection(&hConnectCriticalSect
ion);
                                        }


        DeleteCriticalSection(&TermCriticalSection)
;

                                        if
(hLibInstanceTm != NULL)

        FreeLibrary( hLibInstanceTm );

        hLibInstanceTm = NULL;

                                        if
(hLibInstanceDb != NULL)

        FreeLibrary( hLibInstanceDb );

        hLibInstanceDb = NULL;

                                        Sleep(500);
                                        break;

                                default:
                                        /* nothing
*/;
                }
        }
        catch (CBaseErr *e)
        {
                TCHAR szMsg[256];

                _sntprintf(szMsg, sizeof(szMsg),
"%s error, code %d: %s",
                                        e-
>ErrorTypeStr(), e->ErrorNum(), e->ErrorText());
                WriteMessageToEventLog( szMsg );
                delete e;
                TerminateExtension(0);
                return FALSE;
        }
        catch (...)
        {
                WriteMessageToEventLog(TEXT("Unhandled
exception.  DLL could not load."));
                TerminateExtension(0);
                return FALSE;
        }

        return TRUE;
}


/* FUNCTION: GetExtensionVersion
 *
 * PURPOSE:         This function is called by the
inet service when the DLL is first loaded.
 *
 * ARGUMENTS:       HSE_VERSION_INFO    *pVer
        passed in structure in which to place
expected version number.
 *
 * RETURNS:                 TRUE      inet service
expected return value.
 */

BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO
*pVer)
{
        pVer->dwExtensionVersion =
MAKELONG(HSE_VERSION_MINOR, HSE_VERSION_MAJOR);
        lstrcpyn(pVer->lpszExtensionDesc, "TPC-C
Server.", HSE_MAX_EXT_DLL_NAME_LEN);

        return TRUE;
}

/* FUNCTION: TerminateExtension
 *
 * PURPOSE:         This function is called by the
inet service when the DLL is about to be unloaded.
 *                              Release all resources
in anticipation of being unloaded.
 *
 * RETURNS:                 TRUE      inet service
expected return value.
 */

BOOL WINAPI TerminateExtension( DWORD dwFlags )
{
        if (pDeliHandles)
        {
                SetEvent( hDoneEvent );
                for(DWORD i=0;
i<dwNumDeliveryThreads; i++)
                                WaitForSingleObject(
pDeliHandles[i], INFINITE );
        }

        TermDeleteAll();
        return TRUE;
}


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

DWORD WINAPI
HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
        int                             TermId,
iSyncId;
        char                szBuffer[4096];

        int                             lpbSize;
        static char         szHeader[] = "200 Ok";
        DWORD               dwSize = 6;
        // initial value is strlen(szHeader)
        char                szHeader1[4096];
        DWORD               dwAddr;   // used to
store Win32 exception address
        LPEXCEPTION_POINTERS
        pExceptionInfo;     // pointer to Win32
exception info


#ifdef ICECAP
        StartCAP();
#endif

        // Use structured exception handling for
Win32 exceptions
        //
        __try
        {
                ProcessCommand(pECB, szBuffer,
TermId, iSyncId);
        }
        __except (          pExceptionInfo =
GetExceptionInformation(), // can call
GetExceptionInformation only in filter (not handler)
                                                dwAddr =
(DWORD)pExceptionInfo->ExceptionRecord-
>ExceptionAddress,  // save the address
```

```
        EXCEPTION_EXECUTE_HANDLER)    // handle all
exceptions
        {
                char
        szMsg[512];
                int
        iLen;

                MEMORY_BASIC_INFORMATION mbi ;
                VirtualQuery( (void*)dwAddr,
&mbi, sizeof( mbi ) ) ;
                DWORD hInstance =
(DWORD)mbi.AllocationBase ;

                iLen = wsprintf(szMsg,
TEXT("Unhandled exception (%#x) in Web Client's
HttpExtensionProc. "
                                        "Occured at
address %#x, base %#x, tpcc_com.dll at %#x, tpcc.dll
at %#x, tpcc_com_all.dll at %#x"),

        GetExceptionCode(), dwAddr, hInstance,

        GetModuleHandle("tpcc_com.dll"),
GetModuleHandle("tpcc.dll"),
GetModuleHandle("tpcc_com_all.dll"));

                if (txnDelilog != NULL)
                {
                        txnDelilog-
>WriteCtrlRecToLog(TXN_EVENT_WARNING, szMsg, iLen +
1);

                }
                ErrorForm( pECB, ERR_TYPE_WEBDLL,
GetExceptionCode(), TermId, iSyncId, szMsg, szBuffer
);
        }

#ifdef ICECAP
        StopCAP();
#endif

        lpbSize = strlen(szBuffer);
        dwSize += lpbSize;
        dwSize += wsprintf(szHeader1,
                        "Content-Type:
text/html\r\n"
                        "Content-Length:
%d\r\n"
                        "Connection: Keep-
Alive\r\n\r\n" , lpbSize);
        strcat( szHeader1, szBuffer );

        (*pECB->ServerSupportFunction)(pECB-
>ConnID, HSE_REQ_SEND_RESPONSE_HEADER, szHeader,
(LPDWORD) &dwSize, (LPDWORD)szHeader1);

        //finish up and keep connection
        pECB->dwHttpStatusCode = 200;
        return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}
```

```
/* FUNCTION: ProcessCommand
 *
 * PURPOSE:         This function parses the commands
from the driver and executes corresponding
transactions.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK
                *pECB     structure pointer to passed in
internet
 *

                    service information.
 *
 * RETURNS:                   None (outputs into the
szBuffer parameter).
 *
 * COMMENTS:        Separated from HttpExtensionProc
to be able to use structured exception handling in
 *
         HttpExtensionProc (cannot mix C++ and Win32
exceptions in one functions).
 *
 */
void ProcessCommand(EXTENSION_CONTROL_BLOCK *pECB,
char* szBuffer, int& TermId, int& iSyncId)
{
         int                            iCmd, FormId;

         try
         {
                  //process http query
                  ProcessQueryString(pECB, &iCmd,
&FormId, &TermId, &iSyncId);

                  if (TermId != 0)
                  {
                           if ( TermId < 0 ||
TermId >= Term.iNumEntries ||
Term.pClientData[TermId].iNextFree != -1 )
                           {
                                    //
debugging...
                                    char
szTmp[128];
                                    wsprintf(
szTmp, "Invalid term ID; TermId = %d", TermId );

         WriteMessageToEventLog( szTmp );

                                    throw new
CWEBCLNT_ERR( ERR_INVALID_TERMID );
                           }

                           //must have a valid
syncid here since termid is valid
                           if (iSyncId !=
Term.pClientData[TermId].iSyncId)
                                    throw new
CWEBCLNT_ERR( ERR_INVALID_SYNC_CONNECTION );

                           //set use time
```

```
         Term.pClientData[TermId].iTickCount =
GetTickCount();
                  }


                  switch(iCmd)
                  {
                  case 0:
                           WelcomeForm(pECB,
szBuffer);
                           break;
                  case 1:
                           switch( FormId )
                           {
                           case WELCOME_FORM:
                           case MAIN_MENU_FORM:
                                    break;
                           case NEW_ORDER_FORM:

         ProcessNewOrderForm(pECB, TermId,
szBuffer);
                                    break;
                           case PAYMENT_FORM:

         ProcessPaymentForm(pECB, TermId, szBuffer);
                                    break;
                           case DELIVERY_FORM:

         ProcessDeliveryForm(pECB, TermId,
szBuffer);
                                    break;
                           case ORDER_STATUS_FORM:

         ProcessOrderStatusForm(pECB, TermId,
szBuffer);
                                    break;
                           case STOCK_LEVEL_FORM:

         ProcessStockLevelForm(pECB, TermId,
szBuffer);
                                    break;
                           }
                           break;
                  case 2:
                           // new-order selected
from menu; display new-order input form

         MakeNewOrderForm(TermId, NULL, INPUT_FORM,
szBuffer);
                           break;
                  case 3:
                           // payment selected
from menu; display payment input form
                           MakePaymentForm(TermId,
NULL, INPUT_FORM, szBuffer);
                           break;
                  case 4:
                           // delivery selected
from menu; display delivery input form

         MakeDeliveryForm(TermId, NULL, INPUT_FORM,
szBuffer);
```

```
                        break;
                case 5:
                        // order-status
selected from menu; display order-status input form

        MakeOrderStatusForm(TermId, NULL,
INPUT_FORM, szBuffer);
                        break;
                case 6:
                        // stock-level selected
from menu; display stock-level input form

        MakeStockLevelForm(TermId, NULL,
INPUT_FORM, szBuffer);
                        break;
                case 7:
                        // ExitCmd
                        TermDelete(TermId);
                        WelcomeForm(pECB,
szBuffer);
                        break;
                case 8:
                        SubmitCmd(pECB,
szBuffer);
                        break;
                case 9:
                        // menu

        MakeMainMenuForm(TermId,
Term.pClientData[TermId].iSyncId, szBuffer);
                        break;
                case 10:
                        // CMD=Clear
                        // resets all
connections; should only be used when no other
connections are active
                        TermDeleteAll();
                        TermInit();
                        WelcomeForm(pECB,
szBuffer);
                        break;
                case 11:  // CMD=Stats
                        StatsCmd(pECB,
szBuffer);
                        break;
                }
        }
        catch (CBaseErr *e)
        {
                ErrorForm( pECB, e->ErrorType(),
e->ErrorNum(), TermId, iSyncId, e->ErrorText(),
szBuffer );
                delete e;
        }
}

void WriteMessageToEventLog(LPTSTR lpszMsg)
{
    TCHAR   szMsg[256];
    HANDLE  hEventSource;
    LPTSTR  lpszStrings[2];

    // Use event logging to log the error.
    //
    hEventSource = RegisterEventSource(NULL,
TEXT("TPCC.DLL"));

    _stprintf(szMsg, TEXT("Error in TPCC.DLL: "));
    lpszStrings[0] = szMsg;
    lpszStrings[1] = lpszMsg;

    if (hEventSource != NULL)
          {
        ReportEvent(hEventSource, // handle of event
source
            EVENTLOG_ERROR_TYPE,  // event type
            0,                    // event category
            0,                    // event ID
            NULL,                 // current user's
SID
            2,                    // strings in
lpszStrings
            0,                    // no bytes of raw
data
            (LPCTSTR *)lpszStrings,    // array of
error strings
            NULL);                // no raw data

        (VOID) DeregisterEventSource(hEventSource);
    }
}


/* FUNCTION: DeliveryWorkerThread
 *
 * PURPOSE:         This function processes deferred
delivery txns.  There are typically several
 *                          threads running this
routine.  The number of threads is determined by an
entry
 *                          read from the registry.
The thread waits for work by waiting on semaphore.
 *                          When a delivery txn is
posted, the semaphore is released.  After processing
 *                          the delivery txn,
information is logged to record the txn status and
execution
 *                          time.
 */

/*static*/ void DeliveryWorkerThread(void *ptr)
{
          CTPCC_BASE                    *pTxn = NULL;

          DELIVERY_TRANSACTION
          delivery;
          PDELIVERY_DATA
          pDeliveryData;
          TXN_RECORD_TPCC_DELIV_DEF     txnDeliRec;

          DWORD
          index;
          HANDLE
          handles[2];
          SYSTEMTIME                    trans_end;
                  //delivery transaction finished
time
          SYSTEMTIME                    trans_start;
          //delivery transaction start time

          assert(txnDelilog != NULL);

          try
          {
                  if (Reg.eDB_Protocol == ODBC)
                  {
                          if (Reg.dwConnectDelay
> 0)
                          {
                                  //
Synchronize connect (for VIA)
                                  //

          EnterCriticalSection(&hConnectCriticalSecti
on);


          Sleep(Reg.dwConnectDelay);


          LeaveCriticalSection(&hConnectCriticalSecti
on);
                          }

                          pTxn = pCTPCC_ODBC_new(
Reg.szDbServer, Reg.szDbUser, Reg.szDbPassword,

          szMyComputerName, Reg.szDbName,

          Reg.szSPPrefix,
Reg.bCallNoDuplicatesNewOrder );

                  }
                  pDeliveryData = pTxn-
>BuffAddr_Delivery();
          }
          catch (CBaseErr *e)
          {
                  char szTmp[1024];
                  wsprintf( szTmp, "Error in
Delivery Txn thread.  Could not connect to database.
"
                                          "%s.
Server=%s, User=%s, Password=%s, Database=%s",
                                          e-
>ErrorText(), Reg.szDbServer, Reg.szDbUser,
Reg.szDbPassword, Reg.szDbName );
                  WriteMessageToEventLog( szTmp );
                  delete e;
                  goto ErrorExit;
          }
          catch (...)
          {

          WriteMessageToEventLog(TEXT("Unhandled
exception caught in DeliveryWorkerThread."));
                  goto ErrorExit;
```

```
	}

	while (TRUE)
	{
		try
		{
			//while delivery thread
running, i.e. user has not requested termination
			while (TRUE)
			{
				// need to
wait for multiple objects: program exit or worker
semaphore;
				handles[0] =
hDoneEvent;
				handles[1] =
hWorkerSemaphore;
				index =
WaitForMultipleObjects( 2, &handles[0], FALSE,
INFINITE );
				if (index ==
WAIT_OBJECT_0)

	goto ErrorExit;


	ZeroMemory(&txnDeliRec,
sizeof(txnDeliRec));

	txnDeliRec.TxnType =
TXN_REC_TYPE_TPCC_DELIV_DEF;

				// make a
local copy of current entry from delivery buffer and
increment buffer index

	EnterCriticalSection(&DelBuffCriticalSectio
n);
				delivery =
*(pDelBuff+dwDelBuffBusyIndex);

	dwDelBuffFreeCount++;

	dwDelBuffBusyIndex++;
				if
(dwDelBuffBusyIndex == dwDelBuffSize)	// wrap-
around if at end of buffer

	dwDelBuffBusyIndex = 0;


	LeaveCriticalSection(&DelBuffCriticalSectio
n);


	pDeliveryData->w_id = delivery.w_id;

	pDeliveryData->o_carrier_id =
delivery.o_carrier_id;


	txnDeliRec.w_id = pDeliveryData->w_id;

	txnDeliRec.o_carrier_id = pDeliveryData-
>o_carrier_id;

	txnDeliRec.TxnStartT0 =
Get64BitTime(&delivery.queue);

				GetLocalTime(
&trans_start );
				pTxn-
>Delivery();
				GetLocalTime(
&trans_end );

				//log txn

	txnDeliRec.TxnStatus = ERR_SUCCESS;
				for (int i=0;
i<10; i++)

	txnDeliRec.o_id[i] = pDeliveryData-
>o_id[i];

	txnDeliRec.DeltaT4 =
(int)(Get64BitTime(&trans_end) -
txnDeliRec.TxnStartT0);

	txnDeliRec.DeltaTxnExec =
(int)(Get64BitTime(&trans_end) -
Get64BitTime(&trans_start));

				if
(txnDelilog != NULL)

	txnDelilog->WriteToLog(&txnDeliRec);
				}
			}
			catch (CBaseErr *e)
			{
				char szTmp[1024];
				wsprintf( szTmp, "%s
Error (code %d) in Delivery Txn thread. %s",

e->ErrorTypeStr(), e->ErrorNum(), e->ErrorText() );
				WriteMessageToEventLog(
szTmp );

				// log the error txn
				txnDeliRec.TxnStatus =
e->ErrorType();
				if (txnDelilog != NULL)
					txnDelilog-
>WriteToLog(&txnDeliRec);

				delete e;
			}
			catch (...)
			{
				// unhandled exception;
shouldn't happen; not much we can do...

	WriteMessageToEventLog(TEXT("Unhandled
exception caught in DeliveryWorkerThread."));
			}
		}

ErrorExit:
		if (Reg.dwConnectDelay > 0)
		{
			// Synchronize disconnect (for
VIA)
			//

	EnterCriticalSection(&hConnectCriticalSecti
on);

			Sleep(Reg.dwConnectDelay);
		}

		delete pTxn;

		if (Reg.dwConnectDelay > 0)
		{
			// Synchronize disconnect (for
VIA)
			//

	LeaveCriticalSection(&hConnectCriticalSecti
on);
		}

		_endthread();
}

/* FUNCTION: PostDeliveryInfo
 *
 * PURPOSE:		This function enters the delivery
txn into the deferred delivery buffer.
 *
 * RETURNS:			BOOL	FALSE
		delivery information posted successfully
 *
		TRUE	error cannot post delivery info
 */

BOOL PostDeliveryInfo(long w_id, short o_carrier_id)
{
		BOOL bError;

		EnterCriticalSection(&DelBuffCriticalSectio
n);
		if (dwDelBuffFreeCount > 0)
		{
			bError = FALSE;
			(pDelBuff+dwDelBuffFreeIndex)-
>w_id			= w_id;
			(pDelBuff+dwDelBuffFreeIndex)-
>o_carrier_id	= o_carrier_id;

		GetLocalTime(&(pDelBuff+dwDelBuffFreeIndex)
->queue);

			dwDelBuffFreeCount--;
			dwDelBuffFreeIndex++;
			if (dwDelBuffFreeIndex ==
dwDelBuffSize)
```

```
                        dwDelBuffFreeIndex = 0;
                // wrap-around if at end of
buffer
        }
        else
                // No free buffers.  Return an
error, which indicates that the delivery buffer is
full.
                // Most likely, the number of
delivery worker threads needs to be increased to keep
up
                // with the txn rate.
                bError = TRUE;
        LeaveCriticalSection(&DelBuffCriticalSectio
n);

        if (!bError)
                // increment worker semaphore to
wake up a worker thread
                ReleaseSemaphore(
hWorkerSemaphore, 1, NULL );

        return bError;
}

/* FUNCTION: ProcessQueryString
 *
 * PURPOSE:         This function extracts the
relevent information out of the http command passed
in from
 *                          the browser.
 *
 * COMMENTS:        If this is the initial connection
i.e. client is at welcome screen then
 *                                  there will
not be a terminal id or current form id.  If this is
the case
 *                                  then the
pTermid and pFormid return values are undefined.
 */

void ProcessQueryString(EXTENSION_CONTROL_BLOCK
*pECB, int *pCmd, int *pFormId, int *pTermId, int
*pSyncId)
{
        char *ptr = pECB->lpszQueryString;
        char szBuffer[25];
        int i;

        //allowable client command strings i.e.
CMD=command
        static char *szCmds[] =
        {
                "Process", "..NewOrder..",
"..Payment..", "..Delivery..", "..Order-Status..",
"..Stock-Level..",
                "..Exit..", "Submit", "Menu",
"Clear", "Stats", ""
        };

        *pCmd      = 0;              // default is
the login screen
        *pTermId = 0;

        // if no params (i.e., empty query string),
then return login screen
        if (strlen(pECB->lpszQueryString) == 0)
                return;

        // parse FORMID, TERMID, and SYNCID
        *pFormId = GetIntKeyValue(&ptr, "FORMID",
NO_ERR, NO_ERR);
        *pTermId = GetIntKeyValue(&ptr, "TERMID",
NO_ERR, NO_ERR);
        *pSyncId = GetIntKeyValue(&ptr, "SYNCID",
NO_ERR, NO_ERR);

        // parse CMD
        GetKeyValue(&ptr, "CMD", szBuffer,
sizeof(szBuffer), ERR_COMMAND_UNDEFINED);

        // see which command it matches
        for(i=0; ; i++)
        {
                if (szCmds[i][0] == 0)
                        // no more; no match;
return error
                        throw new CWEBCLNT_ERR(
ERR_COMMAND_UNDEFINED );
                if ( !strcmp(szCmds[i], szBuffer)
)
                {
                        *pCmd = i+1;
                        break;
                }
        }
}

/* FUNCTION: void WelcomeForm
 *
 */

void WelcomeForm(EXTENSION_CONTROL_BLOCK *pECB, char
*szBuffer)
{
        char szTmp[1024];

        //welcome to tpc-c html form buffer, this
is first form client sees.
        strcpy( szBuffer,
        "<HTML><HEAD><TITLE>TPC-C Web
Client</TITLE></HEAD><BODY>"

        "<B><BIG>Microsoft TPC-C Web Client (ver
4.69)</BIG></B> <BR> <BR>"

        "<font face=\"Courier New\"><PRE>"

        "Compiled: "__DATE__", "__TIME__" <BR>"

        "Source:   "__FILE__" ("__TIMESTAMP__")
<BR>"

        "</PRE></font>"

        "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">"

        "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\"
VALUE=\"0\">"

        "<INPUT TYPE=\"hidden\" NAME=\"ERROR\"
VALUE=\"0\">"

        "<INPUT TYPE=\"hidden\" NAME=\"FORMID\"
VALUE=\"1\">"

        "<INPUT TYPE=\"hidden\" NAME=\"TERMID\"
VALUE=\"0\">"

        "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\"
VALUE=\"0\">"

        "<INPUT TYPE=\"hidden\" NAME=\"VERSION\"
VALUE=\"" WEBCLIENT_VERSION "\">"
                        );

        sprintf( szTmp,     "Configuration
Settings: <BR><font face=\"Courier New\"
color=\"blue\"><PRE>"

        "Txn Monitor            = <B>%s</B><BR>"

        "Database protocol      = <B>%s</B><BR>"

        "Max Connections        = <B>%d</B><BR>"
                                                "#
of Delivery Threads  = <B>%d</B><BR>"

        "Max Pending Deliveries = <B>%d</B><BR>"
                        ,
szTxnMonNames[Reg.eTxnMon],
szDBNames[Reg.eDB_Protocol],
                        Reg.dwMaxConnections,
dwNumDeliveryThreads, dwDelBuffSize );
        strcat( szBuffer, szTmp);

        if (Reg.eTxnMon == COM)
        {
                sprintf( szTmp,     "COM Single
Pool        = <B>%s</B><BR>",
                        Reg.bCOM_SinglePool ?
"YES" : "NO" );
                strcat( szBuffer, szTmp);
        }
        strcat( szBuffer, "</PRE></font>");

        if (Reg.eTxnMon == None)
                // connection options may be
specified when not using a txn monitor
                sprintf( szTmp,     "Please enter
your database options for this connection:<BR>"

        "<font face=\"Courier New\"
color=\"blue\"><PRE>"

        "DB Server    = <INPUT NAME=\"db_server\"
SIZE=20 VALUE=\"%s\"><BR>"
```

```
          "DB User ID   = <INPUT NAME=\"db_user\"
SIZE=20 VALUE=\"%s\"><BR>"

          "DB Password  = <INPUT NAME=\"db_passwd\"
SIZE=20 VALUE=\"%s\"><BR>"

          "DB Name      = <INPUT NAME=\"db_name\"
SIZE=20 VALUE=\"%s\"><BR>"

          "</PRE></font>"
                                        ,
Reg.szDbServer, Reg.szDbUser, Reg.szDbPassword,
Reg.szDbName );
          else
                    // if using a txn monitor,
connection options are determined from registry;
can't
                    // set per user.  show options
fyi
                    sprintf( szTmp,     "Database
options which will be used by the transaction
monitor:<BR>"

          "<font face=\"Courier New\"
color=\"blue\"><PRE>"

          "DB Server              = <B>%s</B><BR>"

          "DB User ID             = <B>%s</B><BR>"

          "DB Password            = <B>%s</B><BR>"

          "DB Name                = <B>%s</B><BR>"

          "</PRE></font>"
                                        ,
Reg.szDbServer, Reg.szDbUser, Reg.szDbPassword,
Reg.szDbName );
          strcat( szBuffer, szTmp);

          sprintf( szTmp,     "Please enter your
Warehouse and District for this session:<BR>"

          "<font face=\"Courier New\"
color=\"blue\"><PRE>" );
          strcat( szBuffer, szTmp);
          strcat( szBuffer,   "Warehouse ID = <INPUT
NAME=\"w_id\" SIZE=6><BR>"

          "District ID  = <INPUT NAME=\"d_id\"
SIZE=2><BR>"

          "</PRE></font><HR>"

          "<INPUT TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"Submit\">"

          "</FORM></BODY></HTML>");
}

/* FUNCTION: SubmitCmd
 *
 * PURPOSE:         This function allocated a new
terminal id in the Term structure array.
 *
 */

void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, char
*szBuffer)
{
          int                 iNewTerm;
          char      *ptr = pECB->lpszQueryString;

          char      szVersion[32]       = { 0 };
          char      szServer[32]        = { 0 };
          char      szUser[32]                  =
"sa";
          char      szPassword[32]      = { 0 };
          char      szDatabase[32]      = "tpcc";

          // validate version field; the version
field ensures that the RTE is synchronized with the
web client
          GetKeyValue(&ptr, "VERSION", szVersion,
sizeof(szVersion), ERR_VERSION_MISMATCH);
          if ( strcmp( szVersion, WEBCLIENT_VERSION )
)
                    throw new CWEBCLNT_ERR(
ERR_VERSION_MISMATCH );

          if (Reg.eTxnMon == None)
          {
                    // parse Server name
                    GetKeyValue(&ptr, "db_server",
szServer, sizeof(szServer), ERR_NO_SERVER_SPECIFIED);
                    // parse User name
                    GetKeyValue(&ptr, "db_user",
szUser, sizeof(szUser), NO_ERR);
                    // parse Password
                    GetKeyValue(&ptr, "db_passwd",
szPassword, sizeof(szPassword), NO_ERR);
                    // parse Database name
                    GetKeyValue(&ptr, "db_name",
szDatabase, sizeof(szDatabase), NO_ERR);
          }

          // parse warehouse ID
          int w_id = GetIntKeyValue(&ptr, "w_id",
ERR_HTML_ILL_FORMED, ERR_W_ID_INVALID);
          if ( w_id < 1 )
                    throw new CWEBCLNT_ERR(
ERR_W_ID_INVALID );

          // parse district ID
          int d_id = GetIntKeyValue(&ptr, "d_id",
ERR_HTML_ILL_FORMED, ERR_D_ID_INVALID);
          if ( d_id < 1 || d_id > 10 )
                    throw new CWEBCLNT_ERR(
ERR_D_ID_INVALID );


          iNewTerm = TermAdd();

          Term.pClientData[iNewTerm].w_id = w_id;
          Term.pClientData[iNewTerm].d_id = d_id;
          try
          {
                    if (Reg.eTxnMon == COM)

          Term.pClientData[iNewTerm].pTxn =
pCTPCC_COM_new( Reg.bCOM_SinglePool );
                    else if (Reg.eDB_Protocol ==
ODBC)

          Term.pClientData[iNewTerm].pTxn =
pCTPCC_ODBC_new(    szServer, szUser, szPassword,
szMyComputerName,



          szDatabase, Reg.szSPPrefix,



          Reg.bCallNoDuplicatesNewOrder );
          }
          catch (...)
          {
                    TermDelete(iNewTerm);
                    throw;              // pass
exception upward
          }

          MakeMainMenuForm(iNewTerm,
Term.pClientData[iNewTerm].iSyncId, szBuffer);
}

/* FUNCTION: StatsCmd
 *
 * PURPOSE:         This function returns to the
browser the total number of active terminal ids.
 *                           This routine is for
development/debugging purposes.
 *
 */

void StatsCmd(EXTENSION_CONTROL_BLOCK *pECB, char
*szBuffer)
{
          int i;
          int       iTotal;

          EnterCriticalSection(&TermCriticalSection);

          iTotal = 0;
          for(i=0; i<Term.iNumEntries; i++)
          {
                    if (Term.pClientData[i].iNextFree
== -1)
                              iTotal++;
          }

          LeaveCriticalSection(&TermCriticalSection);

          wsprintf( szBuffer,
```

```
        "<HTML><HEAD><TITLE>TPC-C Web Client
Stats</TITLE></HEAD>"
                        "<BODY><B><BIG> Total
Active Connections: %d </BIG></B><BR></BODY></HTML>"
                        , iTotal );
}

char *CWEBCLNT_ERR::ErrorText()
{
        static SERRORMSG errorMsgs[] =
        {
                {       ERR_COMMAND_UNDEFINED,

        "Command undefined."

                                                },
                {       ERR_D_ID_INVALID,

        "Invalid District ID Must be 1 to 10."

                },
                {
        ERR_DELIVERY_CARRIER_ID_RANGE,
                "Delivery Carrier ID out of range
must be 1 - 10."
        },
                {
        ERR_DELIVERY_CARRIER_INVALID,
        "Delivery Carrier ID invalid must be
numeric 1 - 10."                                },
                {
        ERR_DELIVERY_MISSING_OCD_KEY,
        "Delivery missing Carrier ID key \"OCD*\"."

                },
                {
        ERR_DELIVERY_THREAD_FAILED,
                "Could not start delivery worker
thread."
                },
                {       ERR_GETPROCADDR_FAILED,

        "Could not map proc in DLL.  GetProcAddr
error.  DLL="                                   },
                {       ERR_HTML_ILL_FORMED,

        "Required key field is missing from HTML
string."                                        },
                {
        ERR_INVALID_SYNC_CONNECTION,
        "Invalid Terminal Sync ID."

                                        },
                {       ERR_INVALID_TERMID,

        "Invalid Terminal ID."

                                                },
                {       ERR_LOADDLL_FAILED,

        "Load of DLL failed.  DLL="
```

```
                                        },
                {
        ERR_MAX_CONNECTIONS_EXCEEDED,
        "No connections available.  Max Connections
is probably too low."   },
                {
        ERR_MISSING_REGISTRY_ENTRIES,
        "Required registry entries are missing.
Rerun INSTALL to correct."    },
                {
        ERR_NEWORDER_CUSTOMER_INVALID,
                "New Order customer id invalid
data type, range = 1 to 3000."                  },
                {
        ERR_NEWORDER_CUSTOMER_KEY,
                "New Order missing Customer key
\"CID*\"."
                                },
                {
        ERR_NEWORDER_DISTRICT_INVALID,
                "New Order District ID Invalid
range 1 - 10."
                },
                {
        ERR_NEWORDER_FORM_MISSING_DID,
                "New Order missing District key
\"DID*\"."
                                },
                {
        ERR_NEWORDER_ITEMID_INVALID,
        "New Order Item Id is wrong data type, must
be numeric."                                    },
                {
        ERR_NEWORDER_ITEMID_RANGE,
                "New Order Item Id is out of
range. Range = 1 to 999999."                    },
                {
        ERR_NEWORDER_ITEMID_WITHOUT_SUPPW,
        "New Order Item_Id field entered without a
corresponding Supp_W."          },
                {
        ERR_NEWORDER_MISSING_IID_KEY,
        "New Order missing Item Id key \"IID*\"."

                },
                {
        ERR_NEWORDER_MISSING_QTY_KEY,
        "New Order Missing Qty key \"Qty##*\"."

                },
                {
        ERR_NEWORDER_MISSING_SUPPW_KEY,
                "New Order missing Supp_W key
\"SP##*\"."
                                },
                {
        ERR_NEWORDER_NOITEMS_ENTERED,
        "New Order No order lines entered."

                                },
                {
        ERR_NEWORDER_QTY_INVALID,
```

```
                "New Order Qty invalid must be
numeric range 1 - 99."
        },
                {       ERR_NEWORDER_QTY_RANGE,

        "New Order Qty is out of range. Range = 1
to 99."                                         },
                {
        ERR_NEWORDER_QTY_WITHOUT_SUPPW,
                "New Order Qty field entered
without a corresponding Supp_W."                },
                {
        ERR_NEWORDER_SUPPW_INVALID,
                "New Order Supp_W invalid data
type must be numeric."
        },
                {
        ERR_NO_SERVER_SPECIFIED,
                "No Server name specified."

                                                },
                {
        ERR_ORDERSTATUS_CID_AND_CLT,
        "Order Status Only Customer ID or Last Name
may be entered, not both."    },
                {
        ERR_ORDERSTATUS_CID_INVALID,
        "Order Status Customer ID invalid, range
must be numeric 1 - 3000."    },
                {
        ERR_ORDERSTATUS_CLT_RANGE,
                "Order Status Customer last name
longer than 16 characters."             },
                {
        ERR_ORDERSTATUS_DID_INVALID,
        "Order Status District invalid, value must
be numeric 1 - 10."           },
                {
        ERR_ORDERSTATUS_MISSING_CID_CLT,
        "Order Status Either Customer ID or Last
Name must be entered."                  },
                {
        ERR_ORDERSTATUS_MISSING_CID_KEY,
        "Order Status missing Customer key
\"CID*\"."
                },
                {
        ERR_ORDERSTATUS_MISSING_CLT_KEY,
        "Order Status missing Customer Last Name
key \"CLT*\"."                                  },
                {
        ERR_ORDERSTATUS_MISSING_DID_KEY,
        "Order Status missing District key
\"DID*\"."
                },
                {
        ERR_PAYMENT_CDI_INVALID,
                "Payment Customer district
invalid must be numeric."
        },
                {
        ERR_PAYMENT_CID_AND_CLT,
```

```
                "Payment Only Customer ID or Last
Name may be entered, not both."         },
                {
        ERR_PAYMENT_CUSTOMER_INVALID,
        "Payment Customer data type invalid, must
be numeric."                                    },
                {
        ERR_PAYMENT_CWI_INVALID,
                "Payment Customer Warehouse
invalid, must be numeric."
        },
                {
        ERR_PAYMENT_DISTRICT_INVALID,
        "Payment District ID is invalid, must be 1
- 10."                                          },
                {
        ERR_PAYMENT_HAM_INVALID,
                "Payment Amount invalid data type
must be numeric."
        },
                {         ERR_PAYMENT_HAM_RANGE,

        "Payment Amount out of range, 0 - 9999.99."

                },
                {
        ERR_PAYMENT_LAST_NAME_TO_LONG,
                "Payment Customer last name
longer than 16 characters."
        },
                {
        ERR_PAYMENT_MISSING_CDI_KEY,
        "Payment missing Customer district key
\"CDI*\"."
        },
                {
        ERR_PAYMENT_MISSING_CID_CLT,
        "Payment Either Customer ID or Last Name
must be entered."                       },
                {
        ERR_PAYMENT_MISSING_CID_KEY,
        "Payment missing Customer Key \"CID*\"."

                },
                {
        ERR_PAYMENT_MISSING_CLT_KEY,
        "Payment missing Customer Last Name key
\"CLT*\"."
        },
                {
        ERR_PAYMENT_MISSING_CWI_KEY,
        "Payment missing Customer Warehouse key
\"CWI*\"."
        },
                {
        ERR_PAYMENT_MISSING_DID_KEY,
        "Payment missing District Key \"DID*\"."

                },
                {
        ERR_PAYMENT_MISSING_HAM_KEY,
        "Payment missing Amount key \"HAM*\"."
                },
                {
        ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,
        "Stock Level; missing Threshold key
\"TT*\"."
        },
                {
        ERR_STOCKLEVEL_THRESHOLD_INVALID,
        "Stock Level; Threshold value must be in
the range = 1 - 99."                     },
                {
        ERR_STOCKLEVEL_THRESHOLD_RANGE,
                "Stock Level Threshold out of
range, range must be 1 - 99."                   },
                {         ERR_VERSION_MISMATCH,

        "Invalid version field.  RTE and Web Client
are probably out of sync." },
                {         ERR_W_ID_INVALID,

        "Invalid Warehouse ID."

                                                },
                {         0,

                          ""



                                                }
        };

        char szTmp[256];
        int i = 0;
        while (TRUE)
        {
                if (errorMsgs[i].szMsg[0] == 0)
                {
                          strcpy( szTmp, "Unknown
error number." );
                          break;
                }
                if (m_Error ==
errorMsgs[i].iError)
                {
                          strcpy( szTmp,
errorMsgs[i].szMsg );
                          break;
                }
                i++;
        }

        if (m_szTextDetail)
                strcat( szTmp, m_szTextDetail );
        if (m_SystemErr)
                wsprintf( szTmp+strlen(szTmp), "
Error=%d", m_SystemErr );

        m_szErrorText = new char[strlen(szTmp)+1];
        strcpy( m_szErrorText, szTmp );
        return m_szErrorText;

}
```

```
/* FUNCTION: GetKeyValue
 *
 * PURPOSE:         This function parses a http
formatted string for specific key values.
 *
 * ARGUMENTS:       char
        *pQueryString       http string from client
browser
 *                                       char
                *pKey                           key
value to look for
 *                                       char
                *pValue
        character array into which to place key's
value
 *                                       int
                        iMax
        maximum length of key value array.
 *                                       WEBERROR
        err
        error value to throw
 *
 * RETURNS:                  nothing.
 *
 * ERROR:           if (the pKey value is not found)
then
 *                                                  if
(err == 0)
 *
        return (empty string)
 *
        else
 *
        throw CWEBCLNT_ERR(err)
 *
 * COMMENTS:        http keys are formatted either
KEY=value& or KEY=value\0. This DLL formats
 *                                       TPC-C input
fields in such a manner that the keys can be
extracted in the
 *                                       above manner.
 */

void GetKeyValue(char **pQueryString, char *pKey,
char *pValue, int iMax, WEBERROR err)
{
        char *ptr;

        if ( !(ptr=strstr(*pQueryString, pKey)) )
                goto ErrorExit;
        ptr += strlen(pKey);
        if ( *ptr != '=' )
                goto ErrorExit;
        ptr++;

        iMax--;  // one position is for terminating
null
        while( *ptr && *ptr != '&' && iMax)
        {
                *pValue++ = *ptr++;
                iMax--;
        }
```

```
        *pValue = 0;  // terminating null

        *pQueryString = ptr;
        return;

ErrorExit:
        if (err != NO_ERR)
                throw new CWEBCLNT_ERR( err );
        *pValue = 0;  // return empty result string
}

/* FUNCTION: GetIntKeyValue
 *
 * PURPOSE:         This function parses a http
formatted string for a specific key value.
 *
 * ARGUMENTS:       char
        *pQueryString       http string from client
browser
 *                                      char
                *pKey                           key
value to look for
 *                                      WEBERROR
        NoKeyErr            error value to throw if
key not found
 *                                      WEBERROR
        NotIntErr           error value to throw if
value not numeric
 *
 * RETURNS:                 integer
 *
 * ERROR:           if (the pKey value is not found)
then
 *                                              if
(NoKeyErr != NO_ERR)
 *
        throw CWEBCLNT_ERR(err)
 *
        else
 *
        return 0
 *                                      else if (non-
numeric char found) then
 *                                              if
(NotIntErr != NO_ERR) then
 *
        throw CWEBCLNT_ERR(err)
 *
        else
 *
        return 0
 *
 * COMMENTS:        http keys are formatted either
KEY=value& or KEY=value\0. This DLL formats
 *                                      TPC-C input
fields in such a manner that the keys can be
extracted in the
 *                                      above manner.
 */

int GetIntKeyValue(char **pQueryString, char *pKey,
WEBERROR NoKeyErr, WEBERROR NotIntErr)
{
        char *ptr0;
        char *ptr;

        if ( !(ptr=strstr(*pQueryString, pKey)) )
                goto ErrorNoKey;
        ptr += strlen(pKey);
        if ( *ptr != '=' )
                goto ErrorNoKey;
        ptr++;

        ptr0 = ptr;                     // remember
starting point
        // scan string until a terminator (null or
&) or a non-digit
        while( *ptr && *ptr != '&' && isdigit(*ptr)
)
                ptr++;

        // make sure we stopped scanning for the
right reason
        if ((ptr0 == ptr) || (*ptr && *ptr != '&'))
        {
                if (NotIntErr != NO_ERR)
                        throw new CWEBCLNT_ERR(
NoKeyErr );
                return 0;
        }

        *pQueryString = ptr;
        return atoi(ptr0);

ErrorNoKey:
        if (NoKeyErr != NO_ERR)
                throw new CWEBCLNT_ERR( NoKeyErr
);
        return 0;
}

/* FUNCTION: TermInit
 *
 * PURPOSE:         This function initializes the
client terminal structure; it is called when the
TPCC.DLL
 *                              is first loaded by the
inet service.
 *
 */

void TermInit(void)
{
        EnterCriticalSection(&TermCriticalSection);

        Term.iMasterSyncId  = 1;
        Term.iNumEntries    =
Reg.dwMaxConnections+1;

        Term.pClientData    = NULL;
        Term.pClientData    =
(PCLIENTDATA)malloc(Term.iNumEntries *
sizeof(CLIENTDATA));
        if (Term.pClientData == NULL)
        {
                LeaveCriticalSection(&TermCriticalSection);
                        throw new CWEBCLNT_ERR(
ERR_MEM_ALLOC_FAILED );
        }

        ZeroMemory( Term.pClientData,
Term.iNumEntries * sizeof(CLIENTDATA) );

        Term.iFreeList                =
Term.iNumEntries-1;
        // build free list
        // note: Term.pClientData[0].iNextFree gets
set to -1, which marks it as "in use".
        //       This is intentional, as the zero
entry is used as an anchor and never
        //                   allocated as an actual
terminal.
        for(int i=0; i<Term.iNumEntries; i++)
                Term.pClientData[i].iNextFree =
i-1;

        LeaveCriticalSection(&TermCriticalSection);
}

/* FUNCTION: TermDeleteAll
 *
 * PURPOSE:         This function frees allocated
resources associated with the terminal structure.
 *
 * ARGUMENTS:       none
 *
 * RETURNS:                 None
 *
 * COMMENTS:        This function is called only when
the inet service unloads the TPCC.DLL
 *
 */

void TermDeleteAll(void)
{
        EnterCriticalSection(&TermCriticalSection);

        for(int i=1; i<Term.iNumEntries; i++)
        {
                if (Term.pClientData[i].iNextFree
== -1)
                        delete
Term.pClientData[i].pTxn;
        }

        Term.iFreeList                = 0;
        Term.iNumEntries    = 0;
        if ( Term.pClientData )
                free(Term.pClientData);
        Term.pClientData    = NULL;

        LeaveCriticalSection(&TermCriticalSection);
}

/* FUNCTION: TermAdd
 *
```

```
 * PURPOSE:         This function assigns a terminal
id which is used to identify a client browser.
 *
 * RETURNS:                  int
        assigned terminal id
 *
 */

int TermAdd(void)
{
          DWORD     i;
          int                 iNewTerm, iTickCount;

          if (Term.iNumEntries == 0)
                    return -1;

          EnterCriticalSection(&TermCriticalSection);
          if (Term.iFreeList != 0)
          {         // position is available
                    iNewTerm = Term.iFreeList;
                    Term.iFreeList =
Term.pClientData[iNewTerm].iNextFree;

          Term.pClientData[iNewTerm].iNextFree = -1;
// indicates this position is in use
          }
          else
          {         // no open slots, so find the
slot that hasn't been used in the longest time and
reuse it
                    for(iNewTerm=1, i=1,
iTickCount=0x7FFFFFFF; i<Reg.dwMaxConnections; i++)
                    {
                              if (iTickCount >
Term.pClientData[i].iTickCount)
                              {
                                        iTickCount =
Term.pClientData[i].iTickCount;
                                        iNewTerm = i;
                              }
                    }
                    // if oldest term is less than
one minute old, it probably means that more
connections
                    // are being attempted than were
specified as "Max Connections" at install.  In this
case,
                    // do not bump existing
connection; instead, return error to requestor.
                    if ((GetTickCount() - iTickCount)
< 60000)
                    {

          LeaveCriticalSection(&TermCriticalSection);
                              throw new CWEBCLNT_ERR(
ERR_MAX_CONNECTIONS_EXCEEDED );
                    }
          }

          Term.pClientData[iNewTerm].iTickCount =
GetTickCount();
          Term.pClientData[iNewTerm].iSyncId =
Term.iMasterSyncId++;
          Term.pClientData[iNewTerm].pTxn = NULL;

          LeaveCriticalSection(&TermCriticalSection);
          return iNewTerm;
}

/* FUNCTION: TermDelete
 *
 * PURPOSE:         This function makes a terminal
entry in the Term array available for reuse.
 *
 * ARGUMENTS:       int
                                  id
          Terminal id of client exiting
 *
 */

void TermDelete(int id)
{
          if ( id > 0 && id < Term.iNumEntries )
          {
                    delete Term.pClientData[id].pTxn;

                    // put onto free list

          EnterCriticalSection(&TermCriticalSection);

                    Term.pClientData[id].iNextFree =
Term.iFreeList;
                    Term.iFreeList = id;


          LeaveCriticalSection(&TermCriticalSection);
          }
}


/* FUNCTION: MakeErrorForm
 */

void ErrorForm(EXTENSION_CONTROL_BLOCK *pECB, int
iType, int iErrorNum, int iTermId, int iSyncId, char
*szErrorText, char *szBuffer )
{
          wsprintf(szBuffer,
                              "<HTML><HEAD><TITLE>TPC-C
Error</TITLE></HEAD><BODY>"
                              "<FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"%d\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
                              "<BOLD>An Error
Occurred</BOLD><BR><BR>"
                              "%s"
                              "<BR><BR><HR>"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..NewOrder..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Payment..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Delivery..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Order-Status..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Exit..\">"
                              "</FORM></BODY></HTML>"
                              , iType, iErrorNum,
MAIN_MENU_FORM, iTermId, iSyncId, szErrorText );
}

/* FUNCTION: MakeMainMenuForm
 */

void MakeMainMenuForm(int iTermId, int iSyncId, char
*szForm)
{
          wsprintf(szForm,
                              "<HTML><HEAD><TITLE>TPC-C Main
Menu</TITLE></HEAD><BODY>"
                              "Select Desired
Transaction.<BR><HR>"
                              "<FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"0\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
                              "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..NewOrder..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Payment..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Delivery..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Order-Status..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Exit..\">"
                              "</FORM></BODY></HTML>"
                              , MAIN_MENU_FORM, iTermId,
iSyncId);
}

/* FUNCTION: MakeStockLevelForm
 *
 * PURPOSE:         This function constructs the
Stock Level HTML page.
 *
```

```
 * COMMENTS:        The internal client buffer is
created when the terminal id is assigned and should
not
 *                                      be freed
except when the client terminal id is no longer
needed.
 */

void MakeStockLevelForm(int iTermId, STOCK_LEVEL_DATA
*pStockLevelData, BOOL bInput, char *szForm)
{
          int       c;

          c = wsprintf(szForm,
                    "<HTML><HEAD><TITLE>TPC-C Stock
Level</TITLE></HEAD><FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"0\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
                    "<PRE><font face=\"Courier\">
Stock-Level<BR>"
                    "Warehouse: %6.6d   District:
%2.2d<BR> <BR>",
                    STOCK_LEVEL_FORM, iTermId,
Term.pClientData[iTermId].iSyncId,
                    Term.pClientData[iTermId].w_id,
Term.pClientData[iTermId].d_id);

          if ( bInput )
          {
                    strcpy(szForm+c,
                              "Stock Level Threshold:
<INPUT NAME=\"TT*\" SIZE=2><BR> <BR>"
                              "low stock:
</font><BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR>
<BR>"
                              " <BR> <BR> <BR> <BR>
<BR> <BR> <BR></PRE><HR>"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Process\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Menu\">"
                              "</FORM></HTML>" );
          }
          else
          {
                    wsprintf(szForm+c,
                              "Stock Level Threshold:
%2.2d<BR> <BR>"
                              "low stock:
%3.3d</font> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR>
<BR>"
                              " <BR> <BR> <BR> <BR>
<BR> <BR> <BR> <BR></PRE><HR>"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..NewOrder..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Payment..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Delivery..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Order-Status..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Exit..\">"
                              "</FORM></HTML>"
                              , pStockLevelData-
>threshold, pStockLevelData->low_stock);
          }
}

/* FUNCTION: MakeNewOrderForm
 *
 * COMMENTS:        The internal client buffer is
created when the terminal id is assigned and should
not
 *                                      be freed
except when the client terminal id is no longer
needed.
 */

void MakeNewOrderForm(int iTermId, NEW_ORDER_DATA
*pNewOrderData, BOOL bInput, char *szForm)
{
          int                 i, c;
          BOOL      bValid;
          static    char szBR[] = " <BR> <BR> <BR>
<BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR>
<BR> <BR>";

          if (!bInput)
                    assert( pNewOrderData-
>exec_status_code == eOK || pNewOrderData-
>exec_status_code == eInvalidItem );

          bValid = (bInput || (pNewOrderData-
>exec_status_code == eOK));

          c = wsprintf(szForm,
                    "<HTML><HEAD><TITLE>TPC-C New
Order</TITLE></HEAD><BODY>"
                    "<FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
                    "<PRE><font face=\"Courier\">
New Order<BR>"
                    , bValid ? 0 : ERR_BAD_ITEM_ID,
NEW_ORDER_FORM, iTermId,
Term.pClientData[iTermId].iSyncId);

          if ( bInput )
          {
                    c += wsprintf(szForm+c,
"Warehouse: %6.6d   ", Term.pClientData[iTermId].w_id
);

                    strcpy( szForm+c,
                              "District: <INPUT
NAME=\"DID*\" SIZE=1>
Date:<BR>"
                              "Customer:  <INPUT
NAME=\"CID*\" SIZE=4>   Name:
Credit:     %Disc:<BR>"
                              "Order Number:
Number of Lines:           W_tax:         D_tax:<BR>
<BR>"
                              " Supp_W  Item_Id  Item
Name                Qty  Stock  B/G  Price
Amount<BR>"
                              "  <INPUT
NAME=\"SP00*\" SIZE=4>  <INPUT NAME=\"IID00*\"
SIZE=6>                         <INPUT
NAME=\"Qty00*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP01*\" SIZE=4>  <INPUT NAME=\"IID01*\"
SIZE=6>                         <INPUT
NAME=\"Qty01*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP02*\" SIZE=4>  <INPUT NAME=\"IID02*\"
SIZE=6>                         <INPUT
NAME=\"Qty02*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP03*\" SIZE=4>  <INPUT NAME=\"IID03*\"
SIZE=6>                         <INPUT
NAME=\"Qty03*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP04*\" SIZE=4>  <INPUT NAME=\"IID04*\"
SIZE=6>                         <INPUT
NAME=\"Qty04*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP05*\" SIZE=4>  <INPUT NAME=\"IID05*\"
SIZE=6>                         <INPUT
NAME=\"Qty05*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP06*\" SIZE=4>  <INPUT NAME=\"IID06*\"
SIZE=6>                         <INPUT
NAME=\"Qty06*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP07*\" SIZE=4>  <INPUT NAME=\"IID07*\"
SIZE=6>                         <INPUT
NAME=\"Qty07*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP08*\" SIZE=4>  <INPUT NAME=\"IID08*\"
SIZE=6>                         <INPUT
NAME=\"Qty08*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP09*\" SIZE=4>  <INPUT NAME=\"IID09*\"
SIZE=6>                         <INPUT
NAME=\"Qty09*\" SIZE=1><BR>"
```

```
                              "  <INPUT
NAME=\"SP10*\" SIZE=4>  <INPUT NAME=\"IID10*\"
SIZE=6>                            <INPUT
NAME=\"Qty10*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP11*\" SIZE=4>  <INPUT NAME=\"IID11*\"
SIZE=6>                            <INPUT
NAME=\"Qty11*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP12*\" SIZE=4>  <INPUT NAME=\"IID12*\"
SIZE=6>                            <INPUT
NAME=\"Qty12*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP13*\" SIZE=4>  <INPUT NAME=\"IID13*\"
SIZE=6>                            <INPUT
NAME=\"Qty13*\" SIZE=1><BR>"
                              "  <INPUT
NAME=\"SP14*\" SIZE=4>  <INPUT NAME=\"IID14*\"
SIZE=6>                            <INPUT
NAME=\"Qty14*\" SIZE=1><BR>"
                              "Execution Status:
Total:<BR>"
                              "</font></PRE><HR>"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Process\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Menu\">"
                              "</FORM></HTML>"
                              );
          }
          else
          {
                    c += wsprintf(szForm+c,
"Warehouse: %6.6d   District: %2.2d
Date: ",
                              pNewOrderData->w_id,
                              pNewOrderData->d_id);

                    if ( bValid )
                    {
                              c += wsprintf(szForm+c,
"%2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d",

          pNewOrderData->o_entry_d.day,

          pNewOrderData->o_entry_d.month,

          pNewOrderData->o_entry_d.year,

          pNewOrderData->o_entry_d.hour,

          pNewOrderData->o_entry_d.minute,

          pNewOrderData->o_entry_d.second);
                    }

                    c += wsprintf(szForm+c,
"<BR>Customer:  %4.4d   Name: %-16s   Credit: %-2s
",
                              pNewOrderData->c_id,
pNewOrderData->c_last, pNewOrderData->c_credit);

                    if ( bValid )
                    {
                              c += sprintf(szForm+c,

          "%%Disc: %5.2f            <BR>"

          "Order Number: %8.8d  Number of Lines:
%2.2d       W_tax: %5.2f   D_tax: %5.2f  <BR> <BR>"

          " Supp_W  Item_Id  Item Name
Qty  Stock  B/G  Price    Amount<BR>",

          100.0*pNewOrderData->c_discount,

          pNewOrderData->o_id,

          pNewOrderData->o_ol_cnt,
                                        100.0 *
pNewOrderData->w_tax,
                                        100.0 *
pNewOrderData->d_tax);

                              for(i=0;
i<pNewOrderData->o_ol_cnt; i++)
                              {
                                        c +=
sprintf(szForm+c, "%6.6d   %6.6d   %-24s  %2.2d
%3.3d    %1.1s   $%6.2f  $%7.2f  <BR>",

          pNewOrderData->OL[i].ol_supply_w_id,

          pNewOrderData->OL[i].ol_i_id,

          pNewOrderData->OL[i].ol_i_name,

          pNewOrderData->OL[i].ol_quantity,

          pNewOrderData->OL[i].ol_stock,

          pNewOrderData->OL[i].ol_brand_generic,

          pNewOrderData->OL[i].ol_i_price,

          pNewOrderData->OL[i].ol_amount );
                              }
                    }
                    else
                    {
                              c += wsprintf(szForm+c,
                                        "%Disc:<BR>"
                                        "Order
Number: %8.8d  Number of Lines:              W_tax:
D_tax:<BR> <BR>"
                                        " Supp_W
Item_Id  Item Name                 Qty  Stock  B/G
Price    Amount<BR>"
                                        ,
pNewOrderData->o_id);

                              i = 0;
                    }

                    strncpy( szForm+c, szBR, (15-i)*5
);
                    c += (15-i)*5;

                    if ( bValid )
                              c += sprintf(szForm+c,
"Execution Status: Transaction commited.
Total:  $%8.2f  ",

          pNewOrderData->total_amount);
                    else
                              c += wsprintf(szForm+c,
"Execution Status: Item number is not valid.
Total:");

                    strcpy(szForm+c,
                              "
<BR></font></PRE><HR>"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..NewOrder..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Payment..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Delivery..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Order-Status..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Exit..\">"
                              "</FORM></HTML>"
                              );
          }
}

/* FUNCTION: MakePaymentForm
 *
 * COMMENTS:        The internal client buffer is
created when the terminal id is assigned and should
not
 *                                  be freed
except when the client terminal id is no longer
needed.
 */

void MakePaymentForm(int iTermId, PAYMENT_DATA
*pPaymentData, BOOL bInput, char *szForm)
{
          int c;

          c = wsprintf(szForm,
                    "<HTML><HEAD><TITLE>TPC-C
Payment</TITLE></HEAD><BODY>"
                    "<FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"0\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
```

```
                    "<PRE><font face=\"Courier\">
Payment<BR>"
                    "Date: "
                    , PAYMENT_FORM, iTermId,
Term.pClientData[iTermId].iSyncId);

          if ( !bInput )
          {
                    c += wsprintf(szForm+c, "%2.2d-
%2.2d-%4.4d %2.2d:%2.2d:%2.2d",
                              pPaymentData-
>h_date.day,
                              pPaymentData-
>h_date.month,
                              pPaymentData-
>h_date.year,
                              pPaymentData-
>h_date.hour,
                              pPaymentData-
>h_date.minute,
                              pPaymentData-
>h_date.second);
          }

          if ( bInput )
          {
                    c += wsprintf(szForm+c,
                              "<BR> <BR>Warehouse:
%6.6d"
                              "
District: <INPUT NAME=\"DID*\" SIZE=1><BR> <BR> <BR>
<BR> <BR>"
                              "Customer: <INPUT
NAME=\"CID*\" SIZE=4>"
                              "Cust-Warehouse: <INPUT
NAME=\"CWI*\" SIZE=4>  "
                              "Cust-District: <INPUT
NAME=\"CDI*\" SIZE=1><BR>"
                              "Name:
<INPUT NAME=\"CLT*\" SIZE=16>
Since:<BR>"
                              "
Credit:<BR>"
                              "
Disc:<BR>"
                              "
Phone:<BR> <BR>"
                              "Amount Paid:
$<INPUT NAME=\"HAM*\" SIZE=7>      New Cust-
Balance:<BR>"
                              "Credit Limit:<BR>
<BR>Cust-Data: <BR> <BR> <BR> <BR>
<BR></font></PRE><HR>"
                              "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Process\"><INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Menu\">"
                              "</BODY></FORM></HTML>"
                              ,
Term.pClientData[iTermId].w_id);
          }
          else
          {
                    c += wsprintf(szForm+c,
                              "<BR> <BR>Warehouse:
%6.6d                         District: %2.2d<BR>"
                              "%-20s
%-20s<BR>"
                              "%-20s
%-20s<BR>"
                              "%-20s %-2s %5.5s-%4.4s
%-20s %-2s %5.5s-%4.4s<BR> <BR>"
                              "Customer: %4.4d  Cust-
Warehouse: %6.6d  Cust-District: %2.2d<BR>"
                              "Name:   %-16s %-2s %-
16s     Since:  %2.2d-%2.2d-%4.4d<BR>"
                              "        %-20s
Credit: %-2s<BR>"

                              ,
Term.pClientData[iTermId].w_id, pPaymentData->d_id
                              , pPaymentData-
>w_street_1, pPaymentData->d_street_1
                              , pPaymentData-
>w_street_2, pPaymentData->d_street_2
                              , pPaymentData->w_city,
pPaymentData->w_state, pPaymentData->w_zip,
pPaymentData->w_zip+5
                              , pPaymentData->d_city,
pPaymentData->d_state, pPaymentData->d_zip,
pPaymentData->d_zip+5
                              , pPaymentData->c_id,
pPaymentData->c_w_id,         pPaymentData->c_d_id
                              , pPaymentData-
>c_first, pPaymentData->c_middle, pPaymentData-
>c_last
                              , pPaymentData-
>c_since.day, pPaymentData->c_since.month,
                         pPaymentData->c_since.year
                              , pPaymentData-
>c_street_1, pPaymentData->c_credit
                              );

                    c += sprintf(szForm+c,
                              "        %-20s
%%Disc:  %5.2f<BR>",
                              pPaymentData-
>c_street_2, 100.0*pPaymentData->c_discount);

                              c += wsprintf(szForm+c,
                              "        %-20s %-2s
%5.5s-%4.4s       Phone:  %6.6s-%3.3s-%3.3s-%4.4s<BR>
<BR>",
                              pPaymentData->c_city,
pPaymentData->c_state, pPaymentData->c_zip,
pPaymentData->c_zip+5,
                              pPaymentData->c_phone,
pPaymentData->c_phone+6, pPaymentData->c_phone+9,
pPaymentData->c_phone+12 );

                              c += sprintf(szForm+c,
                              "Amount Paid:
$%7.2f      New Cust-Balance: $%14.2f<BR>"
                              "Credit Limit:
$%13.2f<BR> <BR>"
                              , pPaymentData-
>h_amount, pPaymentData->c_balance
                              , pPaymentData-
>c_credit_lim
                              );

                    if ( pPaymentData->c_credit[0] ==
'B' && pPaymentData->c_credit[1] == 'C' )
                              c += wsprintf(szForm+c,

          "Cust-Data: %-50.50s<BR>           %-
50.50s<BR>           %-50.50s<BR>           %-
50.50s<BR>",

          pPaymentData->c_data, pPaymentData-
>c_data+50, pPaymentData->c_data+100, pPaymentData-
>c_data+150 );
                    else
                              strcpy(szForm+c, "Cust-
Data: <BR> <BR> <BR> <BR>");

                    strcat(szForm,      "
<BR></font></PRE><HR>"

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

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

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

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

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

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

          "</BODY></FORM></HTML>");
          }

}

/* FUNCTION: MakeOrderStatusForm
 *
 * COMMENTS:        The internal client buffer is
created when the terminal id is assigned and should
not
 *                                        be freed
except when the client terminal id is no longer
needed.
 */

void MakeOrderStatusForm(int iTermId,
ORDER_STATUS_DATA *pOrderStatusData, BOOL bInput,
char *szForm)
{
          int                 i, c;
          static char szBR[] = " <BR> <BR> <BR> <BR>
<BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR>
<BR>";
```

```
        c = wsprintf(szForm,
                    "<HTML><HEAD><TITLE>TPC-C Order-
Status</TITLE></HEAD><BODY>"
                    "<FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"0\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
                    "<PRE><font face=\"Courier\">
Order-Status<BR>"
                    "Warehouse: %6.6d   ",
                    ORDER_STATUS_FORM, iTermId,
Term.pClientData[iTermId].iSyncId,
Term.pClientData[iTermId].w_id);

        if ( bInput )
        {
                    strcpy(szForm+c,
                                "District: <INPUT
NAME=\"DID*\" SIZE=1><BR>"
                                "Customer: <INPUT
NAME=\"CID*\" SIZE=4>   Name:
<INPUT NAME=\"CLT*\" SIZE=23><BR>"
                                "Cust-Balance:<BR>
<BR>"
                                "Order-Number:
Entry-Date:                     Carrier-
Number:<BR>"
                                "Supply-W     Item-Id
Qty     Amount      Delivery-Date<BR> <BR> <BR> <BR>
<BR>"
                                " <BR> <BR> <BR> <BR>
<BR> <BR> <BR> <BR> <BR> <BR></font></PRE>"
                                "<HR><INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\"><INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">"
                                "</BODY></FORM></HTML>"
);
        }
        else
        {
                    c += wsprintf(szForm+c,
                                "District: %2.2d<BR>"
                                "Customer: %4.4d
Name: %-16s %-2s %-16s<BR>",
                                pOrderStatusData->d_id,
pOrderStatusData->c_id,
                                pOrderStatusData-
>c_first, pOrderStatusData->c_middle,
pOrderStatusData->c_last);

                    c += sprintf(szForm+c, "Cust-
Balance: $%9.2f<BR> <BR>",
                                pOrderStatusData-
>c_balance);

                    c += wsprintf(szForm+c,
                                "Order-Number: %8.8d
Entry-Date: %2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d
Carrier-Number: %2.2d<BR>"
                                "Supply-W     Item-Id
Qty     Amount      Delivery-Date<BR>",
                                pOrderStatusData->o_id,
                                pOrderStatusData-
>o_entry_d.day,
                                pOrderStatusData-
>o_entry_d.month,
                                pOrderStatusData-
>o_entry_d.year,
                                pOrderStatusData-
>o_entry_d.hour,
                                pOrderStatusData-
>o_entry_d.minute,
                                pOrderStatusData-
>o_entry_d.second,
                                pOrderStatusData-
>o_carrier_id);

                    for(i=0; i< pOrderStatusData-
>o_ol_cnt; i++)
                    {
                                c += sprintf(szForm+c,
"  %6.6d     %6.6d     %2.2d     $%8.2f      %2.2d-
%2.2d-%4.4d<BR>",

          pOrderStatusData->OL[i].ol_supply_w_id,

          pOrderStatusData->OL[i].ol_i_id,

          pOrderStatusData->OL[i].ol_quantity,

          pOrderStatusData->OL[i].ol_amount,

          pOrderStatusData->OL[i].ol_delivery_d.day,

          pOrderStatusData-
>OL[i].ol_delivery_d.month,

          pOrderStatusData-
>OL[i].ol_delivery_d.year);
                    }

                    strncpy( szForm+c, szBR, (15-i)*5
);
                    c += (15-i)*5;

                    strcpy(szForm+c,

          "</font></PRE><HR><INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..NewOrder..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Payment..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Delivery..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Order-Status..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Exit..\">"
                                "</BODY></FORM></HTML>"
);
        }
}

/* FUNCTION: MakeDeliveryForm
 *
 * COMMENTS:        The internal client buffer is
created when the terminal id is assigned and should
not
 *                                      be freed
except when the client terminal id is no longer
needed.
 */

void MakeDeliveryForm(int iTermId, DELIVERY_DATA
*pDeliveryData, BOOL bInput, char *szForm)
{
        int       c;

        c = wsprintf(szForm,
                    "<HTML><HEAD><TITLE>TPC-C
Delivery</TITLE></HEAD><BODY>"
                    "<FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
                    "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
                    "<PRE><font face=\"Courier\">
Delivery<BR>"
                    "Warehouse: %6.6d<BR> <BR>",
                    (!bInput && (pDeliveryData-
>exec_status_code != eOK)) ? ERR_TYPE_DELIVERY_POST :
0,
                    DELIVERY_FORM, iTermId,
Term.pClientData[iTermId].iSyncId,
Term.pClientData[iTermId].w_id);

        if ( bInput )
        {
                    strcpy( szForm+c,
                                "Carrier Number: <INPUT
NAME=\"OCD*\" SIZE=1><BR> <BR>"
                                "Execution Status: <BR>
<BR> <BR> <BR> <BR> <BR> <BR> <BR>"
                                " <BR> <BR> <BR> <BR>
<BR> <BR> <BR> <BR> </font></PRE><HR>"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Process\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Menu\">"
                                "</BODY></FORM></HTML>"
);
        }
```

```
        else
        {
                wsprintf( szForm+c,
                                "Carrier Number:
%2.2d<BR> <BR>"
                                "Execution Status: %s
<BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR>"
                                " <BR> <BR> <BR> <BR>
<BR> <BR> <BR> <BR> </font></PRE>"
                                "<HR><INPUT
TYPE=\"submit\" NAME=\"CMD\" VALUE=\"..NewOrder..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Payment..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Delivery..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Order-Status..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Stock-Level..\">"
                                "<INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"..Exit..\">"
                                "</BODY></FORM></HTML>"

                                , pDeliveryData-
>o_carrier_id,
                                (pDeliveryData-
>exec_status_code == eOK) ? "Delivery has been
queued." : "Delivery Post Failed     "
                                );
        }
}

/* FUNCTION: ProcessNewOrderForm
 *
 * PURPOSE:         This function gets and validates
the input data from the new order form
 *                              filling in the required
input variables. it then calls the SQLNewOrder
 *                              transaction, constructs
the output form and writes it back to client
 *                              browser.
 */

void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK
        *pECB, int iTermId, char *szBuffer)
{
        PNEW_ORDER_DATA               pNewOrder;

        pNewOrder = Term.pClientData[iTermId].pTxn-
>BuffAddr_NewOrder();

        ZeroMemory(pNewOrder,
sizeof(NEW_ORDER_DATA));
        pNewOrder->w_id =
Term.pClientData[iTermId].w_id;
        GetNewOrderData(pECB->lpszQueryString,
pNewOrder);

        Term.pClientData[iTermId].pTxn->NewOrder();

        pNewOrder = Term.pClientData[iTermId].pTxn-
>BuffAddr_NewOrder();
        MakeNewOrderForm(iTermId, pNewOrder,
OUTPUT_FORM, szBuffer );
}

/* FUNCTION: void ProcessPaymentForm
 *
 * PURPOSE:         This function gets and validates
the input data from the payment form
 *                              filling in the required
input variables. It then calls the SQLPayment
 *                              transaction, constructs
the output form and writes it back to client
 *                              browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK
        *pECB     passed in structure pointer from
inetsrv.
 *                                      int

        iTermId   client browser terminal id
 *
 */

void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, char *szBuffer)
{
        PPAYMENT_DATA       pPayment;

        pPayment = Term.pClientData[iTermId].pTxn-
>BuffAddr_Payment();
        ZeroMemory(pPayment, sizeof(PAYMENT_DATA));
        pPayment->w_id =
Term.pClientData[iTermId].w_id;
        GetPaymentData(pECB->lpszQueryString,
pPayment);

        Term.pClientData[iTermId].pTxn->Payment();

        pPayment = Term.pClientData[iTermId].pTxn-
>BuffAddr_Payment();
        MakePaymentForm(iTermId, pPayment,
OUTPUT_FORM, szBuffer);
}

/* FUNCTION: ProcessOrderStatusForm
 *
 * PURPOSE:         This function gets and validates
the input data from the Order Status
 *                              form filling in the
required input variables. It then calls the
 *                              SQLOrderStatus
transaction, constructs the output form and writes it
 *                              back to client browser.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK
        *pECB     passed in structure pointer from
inetsrv.
 *                                      int

        iTermId   client browser terminal id
 *
 */

void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, char *szBuffer)
{
        PORDER_STATUS_DATA  pOrderStatus;

        pOrderStatus =
Term.pClientData[iTermId].pTxn-
>BuffAddr_OrderStatus();
        ZeroMemory(pOrderStatus,
sizeof(ORDER_STATUS_DATA));
        pOrderStatus->w_id =
Term.pClientData[iTermId].w_id;
        GetOrderStatusData(pECB->lpszQueryString,
pOrderStatus);

        Term.pClientData[iTermId].pTxn-
>OrderStatus();

        pOrderStatus =
Term.pClientData[iTermId].pTxn-
>BuffAddr_OrderStatus();
        MakeOrderStatusForm(iTermId, pOrderStatus,
OUTPUT_FORM, szBuffer);
}

/* FUNCTION: ProcessDeliveryForm
 *
 * PURPOSE:         This function gets and validates
the input data from the delivery form
 *                              filling in the required
input variables. It then calls the PostDeliveryInfo
 *                              Api, The client is then
informed that the transaction has been posted.
 *
 * ARGUMENTS:       EXTENSION_CONTROL_BLOCK
        *pECB     passed in structure pointer from
inetsrv.
 *                                      int

        iTermId   client browser terminal id
 *
 */

void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, char *szBuffer)
{
        char      *ptr = pECB->lpszQueryString;

        PDELIVERY_DATA      pDelivery;

        pDelivery = Term.pClientData[iTermId].pTxn-
>BuffAddr_Delivery();
        ZeroMemory(pDelivery,
sizeof(DELIVERY_DATA));
        pDelivery->w_id =
Term.pClientData[iTermId].w_id;

        pDelivery->o_carrier_id       =
GetIntKeyValue(&ptr, "OCD*",
ERR_DELIVERY_MISSING_OCD_KEY,
ERR_DELIVERY_CARRIER_INVALID);
        if ( pDelivery->o_carrier_id > 10 ||
pDelivery->o_carrier_id < 1 )
```

```
                    throw new CWEBCLNT_ERR(
ERR_DELIVERY_CARRIER_ID_RANGE );

          if (dwNumDeliveryThreads)
          {
                    //post delivery info
                    if ( PostDeliveryInfo(pDelivery-
>w_id, pDelivery->o_carrier_id) )
                              pDelivery-
>exec_status_code = eDeliveryFailed;
                    else
                              pDelivery-
>exec_status_code = eOK;
          }
          else // delivery is done synchronously if
no delivery threads configured
                    Term.pClientData[iTermId].pTxn-
>Delivery();

          pDelivery = Term.pClientData[iTermId].pTxn-
>BuffAddr_Delivery();
          MakeDeliveryForm(iTermId, pDelivery,
OUTPUT_FORM, szBuffer);
}

/* FUNCTION: ProcessStockLevelForm
 *
 * PURPOSE:          This function gets and validates
the input data from the Stock Level
 *                   form filling in the
required input variables. It then calls the
 *                   SQLStockLevel
transaction, constructs the output form and writes it
 *                   back to client browser.
 *
 * ARGUMENTS:        EXTENSION_CONTROL_BLOCK
          *pECB     passed in structure pointer from
inetsrv.
 *                                   int

          iTermId   client browser terminal id
 *
 */

void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, char *szBuffer)
{
          char                *ptr = pECB-
>lpszQueryString;

          PSTOCK_LEVEL_DATA   pStockLevel;

          pStockLevel =
Term.pClientData[iTermId].pTxn-
>BuffAddr_StockLevel();
          ZeroMemory( pStockLevel,
sizeof(STOCK_LEVEL_DATA) );

          pStockLevel->w_id =
Term.pClientData[iTermId].w_id;
          pStockLevel->d_id =
Term.pClientData[iTermId].d_id;
          pStockLevel->threshold =
GetIntKeyValue(&ptr, "TT*",
ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,
ERR_STOCKLEVEL_THRESHOLD_INVALID);
          if ( pStockLevel->threshold >= 100 ||
pStockLevel->threshold < 0 )
                    throw new CWEBCLNT_ERR(
ERR_STOCKLEVEL_THRESHOLD_RANGE );

          Term.pClientData[iTermId].pTxn-
>StockLevel();

          pStockLevel =
Term.pClientData[iTermId].pTxn-
>BuffAddr_StockLevel();
          MakeStockLevelForm(iTermId, pStockLevel,
OUTPUT_FORM, szBuffer);
}

/* FUNCTION: GetNewOrderData
 *
 * PURPOSE:          This function extracts and
validates the new order form data from an http
command string.
 *
 * ARGUMENTS:        LPSTR
          lpszQueryString               client
browser http command string
 *
          NEW_ORDER_DATA      *pNewOrderData
          pointer to new order data structure
 *
 */

void GetNewOrderData(LPSTR lpszQueryString,
NEW_ORDER_DATA *pNewOrderData)
{
          char      szTmp[26];
          int                 i;
          short     items;
          int                 ol_i_id, ol_quantity;
          char      *ptr = lpszQueryString;

          static char szSP[MAX_OL_NEW_ORDER_ITEMS][6]
=
                    { "SP00*", "SP01*", "SP02*",
"SP03*", "SP04*",
                      "SP05*", "SP06*", "SP07*",
"SP08*", "SP09*",
                      "SP10*", "SP11*", "SP12*",
"SP13*", "SP14*" };
          static char
szIID[MAX_OL_NEW_ORDER_ITEMS][7] =
                    { "IID00*", "IID01*", "IID02*",
"IID03*", "IID04*",
                      "IID05*", "IID06*", "IID07*",
"IID08*", "IID09*",
                      "IID10*", "IID11*", "IID12*",
"IID13*", "IID14*" };
          static char
szQty[MAX_OL_NEW_ORDER_ITEMS][7] =
                    { "Qty00*", "Qty01*", "Qty02*",
"Qty03*", "Qty04*",
                      "Qty05*", "Qty06*", "Qty07*",
"Qty08*", "Qty09*",
                      "Qty10*", "Qty11*", "Qty12*",
"Qty13*", "Qty14*" };

          pNewOrderData->d_id = GetIntKeyValue(&ptr,
"DID*", ERR_NEWORDER_FORM_MISSING_DID,
ERR_NEWORDER_DISTRICT_INVALID);
          pNewOrderData->c_id = GetIntKeyValue(&ptr,
"CID*", ERR_NEWORDER_CUSTOMER_KEY,
ERR_NEWORDER_CUSTOMER_INVALID);

          for(i=0, items=0; i<MAX_OL_NEW_ORDER_ITEMS;
i++)
          {
                    GetKeyValue(&ptr, szSP[i], szTmp,
sizeof(szTmp), ERR_NEWORDER_MISSING_SUPPW_KEY);
                    if ( szTmp[0] )
                    {
                              if ( !IsNumeric(szTmp)
)
                                        throw new
CWEBCLNT_ERR( ERR_NEWORDER_SUPPW_INVALID );
                              pNewOrderData-
>OL[items].ol_supply_w_id = atoi(szTmp);

                              ol_i_id =
pNewOrderData->OL[items].ol_i_id =

          GetIntKeyValue(&ptr, szIID[i],
ERR_NEWORDER_MISSING_IID_KEY,
ERR_NEWORDER_ITEMID_INVALID);
                              if ( ol_i_id > 999999
|| ol_i_id < 1 )
                                        throw new
CWEBCLNT_ERR( ERR_NEWORDER_ITEMID_RANGE );

                              ol_quantity =
pNewOrderData->OL[items].ol_quantity =

          GetIntKeyValue(&ptr, szQty[i],
ERR_NEWORDER_MISSING_QTY_KEY,
ERR_NEWORDER_QTY_INVALID);
                              if ( ol_quantity > 99
|| ol_quantity < 1 )
                                        throw new
CWEBCLNT_ERR( ERR_NEWORDER_QTY_RANGE );

                              items++;
                    }
                    else
                    {         // nothing entered for
supply warehouse, so item id and qty must also be
blank
                              GetKeyValue(&ptr,
szIID[i], szTmp, sizeof(szTmp),
ERR_NEWORDER_MISSING_IID_KEY);
                              if ( szTmp[0] )
                                        throw new
CWEBCLNT_ERR( ERR_NEWORDER_ITEMID_WITHOUT_SUPPW );
```

```
                        GetKeyValue(&ptr,
szQty[i], szTmp, sizeof(szTmp),
ERR_NEWORDER_MISSING_QTY_KEY);
                        if ( szTmp[0] )
                                throw new
CWEBCLNT_ERR( ERR_NEWORDER_QTY_WITHOUT_SUPPW );
                }
        }
        if ( items == 0 )
                throw new CWEBCLNT_ERR(
ERR_NEWORDER_NOITEMS_ENTERED );

        pNewOrderData->o_ol_cnt = items;
}

/* FUNCTION: GetPaymentData
 *
 * PURPOSE:         This function extracts and
validates the payment form data from an http command
string.
 *
 * ARGUMENTS:       LPSTR
        lpszQueryString              client
browser http command string
 *                                   PAYMENT_DATA
        *pPaymentData                pointer to
payment data structure
 */

void GetPaymentData(LPSTR lpszQueryString,
PAYMENT_DATA *pPaymentData)
{
        char    szTmp[26];
        char    *ptr = lpszQueryString;
        BOOL    bCustIdBlank;
        int             iLen;

        pPaymentData->d_id = GetIntKeyValue(&ptr,
"DID*", ERR_PAYMENT_MISSING_DID_KEY,
ERR_PAYMENT_DISTRICT_INVALID);

        GetKeyValue(&ptr, "CID*", szTmp,
sizeof(szTmp), ERR_PAYMENT_MISSING_CID_KEY);
        if ( szTmp[0] == 0 )
        {
                bCustIdBlank = TRUE;
                pPaymentData->c_id = 0;
        }
        else
        {       // parse customer id and verify
that last name was NOT entered
                bCustIdBlank = FALSE;
                if ( !IsNumeric(szTmp) )
                        throw new CWEBCLNT_ERR(
ERR_PAYMENT_CUSTOMER_INVALID );
                pPaymentData->c_id = atoi(szTmp);
        }

        pPaymentData->c_w_id = GetIntKeyValue(&ptr,
"CWI*", ERR_PAYMENT_MISSING_CWI_KEY,
ERR_PAYMENT_CWI_INVALID);

        pPaymentData->c_d_id = GetIntKeyValue(&ptr,
"CDI*", ERR_PAYMENT_MISSING_CDI_KEY,
ERR_PAYMENT_CDI_INVALID);

        if ( bCustIdBlank )
        {       // customer id is blank, so last
name must be entered
                GetKeyValue(&ptr, "CLT*", szTmp,
sizeof(szTmp), ERR_PAYMENT_MISSING_CLT_KEY);
                if ( szTmp[0] == 0 )
                        throw new CWEBCLNT_ERR(
ERR_PAYMENT_MISSING_CID_CLT );

                _strupr( szTmp );
                if ( strlen(szTmp) >
LAST_NAME_LEN )
                        throw new CWEBCLNT_ERR(
ERR_PAYMENT_LAST_NAME_TO_LONG );

                strcpy(pPaymentData->c_last,
szTmp);
                // pad with spaces so that the
client layer doesn't have to do it
                // before passing parameters to
stored procedure
                iLen = strlen(pPaymentData-
>c_last);
                memset(pPaymentData->c_last +
iLen, ' ', LAST_NAME_LEN - iLen);
                pPaymentData-
>c_last[LAST_NAME_LEN] = 0;
        }
        else
        {       // parse customer id and verify
that last name was NOT entered
                GetKeyValue(&ptr, "CLT*", szTmp,
sizeof(szTmp), ERR_PAYMENT_MISSING_CLT_KEY);
                if ( szTmp[0] != 0 )
                        throw new CWEBCLNT_ERR(
ERR_PAYMENT_CID_AND_CLT );
        }

        GetKeyValue(&ptr, "HAM*", szTmp,
sizeof(szTmp), ERR_PAYMENT_MISSING_HAM_KEY);
        if (!IsDecimal(szTmp))
                throw new CWEBCLNT_ERR(
ERR_PAYMENT_HAM_INVALID );
        pPaymentData->h_amount = atof(szTmp);
        if ( pPaymentData->h_amount >= 10000.00 ||
pPaymentData->h_amount < 0 )
                throw new CWEBCLNT_ERR(
ERR_PAYMENT_HAM_RANGE );
}

/* FUNCTION: GetOrderStatusData
 *
 * PURPOSE:         This function extracts and
validates the payment form data from an http command
string.
 *
 */
void GetOrderStatusData(LPSTR lpszQueryString,
ORDER_STATUS_DATA *pOrderStatusData)
{
        char    szTmp[26];
        char    *ptr = lpszQueryString;
        int             iLen;

        pOrderStatusData->d_id =
GetIntKeyValue(&ptr, "DID*",
ERR_ORDERSTATUS_MISSING_DID_KEY,
ERR_ORDERSTATUS_DID_INVALID);

        GetKeyValue(&ptr, "CID*", szTmp,
sizeof(szTmp), ERR_ORDERSTATUS_MISSING_CID_KEY);
        if ( szTmp[0] == 0 )
        {       // customer id is blank, so last
name must be entered
                pOrderStatusData->c_id = 0;
                GetKeyValue(&ptr, "CLT*", szTmp,
sizeof(szTmp), ERR_ORDERSTATUS_MISSING_CLT_KEY);
                if ( szTmp[0] == 0 )
                        throw new CWEBCLNT_ERR(
ERR_ORDERSTATUS_MISSING_CID_CLT );

                _strupr( szTmp );
                if ( strlen(szTmp) >
LAST_NAME_LEN )
                        throw new CWEBCLNT_ERR(
ERR_ORDERSTATUS_CLT_RANGE );

                strcpy(pOrderStatusData->c_last,
szTmp);
                // pad with spaces so that the
client layer doesn't have to do it
                // before passing parameters to
stored procedure
                iLen = strlen(pOrderStatusData-
>c_last);
                memset(pOrderStatusData->c_last +
iLen, ' ', LAST_NAME_LEN - iLen);
                pOrderStatusData-
>c_last[LAST_NAME_LEN] = 0;
        }
        else
        {       // parse customer id and verify
that last name was NOT entered
                if ( !IsNumeric(szTmp) )
                        throw new CWEBCLNT_ERR(
ERR_ORDERSTATUS_CID_INVALID );
                pOrderStatusData->c_id =
atoi(szTmp);
                GetKeyValue(&ptr, "CLT*", szTmp,
sizeof(szTmp), ERR_ORDERSTATUS_MISSING_CLT_KEY);
                if ( szTmp[0] != 0 )
                        throw new CWEBCLNT_ERR(
ERR_ORDERSTATUS_CID_AND_CLT );
        }

}

/* FUNCTION: BOOL IsNumeric(char *ptr)
 *
 * PURPOSE:         This function determines if a
string is numeric. It fails if any characters other
```

```
 *                          than numeric and null
terminator are present.
 *
 * ARGUMENTS:       char
        *ptr      pointer to string to check.
 *
 * RETURNS:                 BOOL      FALSE     if
string is not all numeric
 *
         TRUE      if string contains only numeric
characters i.e. '0' - '9'
 */

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

         while( *ptr && isdigit(*ptr) )
                  ptr++;
         return ( !*ptr );
}

/* FUNCTION: BOOL IsDecimal(char *ptr)
 *
 * PURPOSE:         This function determines if a
string is a non-negative decimal value.
 *          It fails if any characters other than a
series of numbers followed by
 *                          a decimal point,
another series of numbers, and a null terminator are
present.
 *
 * ARGUMENTS:       char
        *ptr      pointer to string to check.
 *
 * RETURNS:                 BOOL      FALSE     if
string is not a valid non-negative decimal value
 *
         TRUE      if string is OK
 */

BOOL IsDecimal(char *ptr)
{
         char *dotptr;
         BOOL  bValid;

         if ( *ptr == 0 )
                  return FALSE;

         // find decimal point
         dotptr = strchr( ptr, '.' );
         if (dotptr == NULL)
                  // no decimal point, so just
check for numeric
                  return IsNumeric(ptr);
         *dotptr = 0;  // temporarily replace
decimal with a terminator

         if ( *ptr != 0 )
                  bValid = IsNumeric(ptr);
         // string starts with decimal point
         else if (*(dotptr+1) == 0)
                  return FALSE;  // nothing but a
decimal point is bad
         else
                  bValid = TRUE;

         if (*(dotptr+1) != 0)
                  // check text after decimal point
                  bValid &= IsNumeric(dotptr+1);

         *dotptr = '.';  // replace decimal point
         return bValid;
}
```

tpcc.def

```
LIBRARY TPCC.DLL

EXPORTS

         GetExtensionVersion @1
         HttpExtensionProc   @2
         TerminateExtension  @3
```

tpcc.h

```
/*       FILE:               TPCC.H
 *                                        Microsoft
TPC-C Kit Ver. 4.69.000
 *                                        Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                                        Version
4.10.000 audited by Richard Gimarc, Performance
Metrics, 3/17/99
 *
 *       PURPOSE:  Header file for ISAPI TPCC.DLL,
defines structures and functions used in the isapi
tpcc.dll.
 *
 */

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

#define TP_MAX_RETRIES
                                        50
```

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

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


//This structure defines the data necessary to keep
distinct for each terminal or client connection.
typedef struct _CLIENTDATA
{
         int                             iNextFree;
                                         //index of
next free element or -1 if this entry in use.
         int                             w_id;
                                         //warehouse
id assigned at welcome form
         int                             d_id;
                                         //district id
assigned at welcome form

         int                             iSyncId;
                                 //syncronization id
         int                             iTickCount;
                                         //time of
last access;

         CTPCC_BASE          *pTxn;

} CLIENTDATA, *PCLIENTDATA;


//This structure is used to define the operational
interface for terminal id support
typedef struct _TERM
{
```

```
        int                             iNumEntries;

        //total allocated terminal array entries
        int                             iFreeList;

        //next available terminal array element or
-1 if none
        int
        iMasterSyncId;
                //syncronization id
        CLIENTDATA          *pClientData;
                                        //pointer to
allocated client data
} TERM;

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


        enum WEBERROR
        {
                NO_ERR,
                ERR_COMMAND_UNDEFINED,
                ERR_D_ID_INVALID,
                ERR_DELIVERY_CARRIER_ID_RANGE,
                ERR_DELIVERY_CARRIER_INVALID,
                ERR_DELIVERY_MISSING_OCD_KEY,
                ERR_DELIVERY_THREAD_FAILED,
                ERR_GETPROCADDR_FAILED,
                ERR_HTML_ILL_FORMED,
                ERR_INVALID_SYNC_CONNECTION,
                ERR_INVALID_TERMID,
                ERR_LOADDLL_FAILED,
                ERR_MAX_CONNECTIONS_EXCEEDED,
                ERR_MEM_ALLOC_FAILED,
                ERR_MISSING_REGISTRY_ENTRIES,
                ERR_NEWORDER_CUSTOMER_INVALID,
                ERR_NEWORDER_CUSTOMER_KEY,
                ERR_NEWORDER_DISTRICT_INVALID,
                ERR_NEWORDER_FORM_MISSING_DID,
                ERR_NEWORDER_ITEMID_INVALID,
                ERR_NEWORDER_ITEMID_RANGE,

        ERR_NEWORDER_ITEMID_WITHOUT_SUPPW,
                ERR_NEWORDER_MISSING_IID_KEY,
                ERR_NEWORDER_MISSING_QTY_KEY,
                ERR_NEWORDER_MISSING_SUPPW_KEY,
                ERR_NEWORDER_NOITEMS_ENTERED,
                ERR_NEWORDER_QTY_INVALID,
                ERR_NEWORDER_QTY_RANGE,
                ERR_NEWORDER_QTY_WITHOUT_SUPPW,
                ERR_NEWORDER_SUPPW_INVALID,
                ERR_NO_SERVER_SPECIFIED,
                ERR_ORDERSTATUS_CID_AND_CLT,
                ERR_ORDERSTATUS_CID_INVALID,
                ERR_ORDERSTATUS_CLT_RANGE,
                ERR_ORDERSTATUS_DID_INVALID,
                ERR_ORDERSTATUS_MISSING_CID_CLT,
                ERR_ORDERSTATUS_MISSING_CID_KEY,
                ERR_ORDERSTATUS_MISSING_CLT_KEY,
                ERR_ORDERSTATUS_MISSING_DID_KEY,
                ERR_PAYMENT_CDI_INVALID,
                ERR_PAYMENT_CID_AND_CLT,
                ERR_PAYMENT_CUSTOMER_INVALID,
                ERR_PAYMENT_CWI_INVALID,
                ERR_PAYMENT_DISTRICT_INVALID,
                ERR_PAYMENT_HAM_INVALID,
                ERR_PAYMENT_HAM_RANGE,
                ERR_PAYMENT_LAST_NAME_TO_LONG,
                ERR_PAYMENT_MISSING_CDI_KEY,
                ERR_PAYMENT_MISSING_CID_CLT,
                ERR_PAYMENT_MISSING_CID_KEY,
                ERR_PAYMENT_MISSING_CLT,
                ERR_PAYMENT_MISSING_CLT_KEY,
                ERR_PAYMENT_MISSING_CWI_KEY,
                ERR_PAYMENT_MISSING_DID_KEY,
                ERR_PAYMENT_MISSING_HAM_KEY,

        ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,
                ERR_STOCKLEVEL_THRESHOLD_INVALID,
                ERR_STOCKLEVEL_THRESHOLD_RANGE,
                ERR_VERSION_MISMATCH,
                ERR_W_ID_INVALID
        };

class CWEBCLNT_ERR : public CBaseErr
{
        public:

                CWEBCLNT_ERR(WEBERROR Err)
                {
                        m_Error = Err;
                        m_szTextDetail = NULL;
                        m_SystemErr = 0;
                        m_szErrorText = NULL;
                };

                CWEBCLNT_ERR(WEBERROR Err, char
*szTextDetail, DWORD dwSystemErr)
                {
                        m_Error = Err;
                        m_szTextDetail = new
char[strlen(szTextDetail)+1];
                        strcpy( m_szTextDetail,
szTextDetail );
                        m_SystemErr =
dwSystemErr;
                        m_szErrorText = NULL;
                };

                ~CWEBCLNT_ERR()
                {
                        if (m_szTextDetail !=
NULL)
                                delete []
m_szTextDetail;
                        if (m_szErrorText !=
NULL)
                                delete []
m_szErrorText;
                };

                WEBERROR  m_Error;
                char
        *m_szTextDetail;    //
                char
        *m_szErrorText;
                DWORD               m_SystemErr;

                int ErrorType() {return
ERR_TYPE_WEBDLL;};
                char *ErrorTypeStr() { return
"WEBCLIENT"; }
                int ErrorNum() {return m_Error;};
                char *ErrorText();

};

//These constants have already been defined in
engstut.h, but since we do
//not want to include it in the delisrv executable
#define   TXN_EVENT_START               2
#define   TXN_EVENT_STOP                4
#define TXN_EVENT_WARNING     6
         //used to record a warning into the log

//function prototypes

BOOL APIENTRY DllMain(HANDLE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved);
void WriteMessageToEventLog(LPTSTR lpszMsg);
void ProcessQueryString(EXTENSION_CONTROL_BLOCK
*pECB, int *pCmd, int *pFormId, int *pTermId, int
*pSyncId);
void WelcomeForm(EXTENSION_CONTROL_BLOCK *pECB, char
*szBuffer);
void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, char
*szBuffer);
void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId);
void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId);
void StatsCmd(EXTENSION_CONTROL_BLOCK *pECB, char
*szBuffer);
void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int
iError, int iErrorType, char *szMsg, int iTermId);
void GetKeyValue(char **pQueryString, char *pKey,
char *pValue, int iMax, WEBERROR err);
int  GetIntKeyValue(char **pQueryString, char *pKey,
WEBERROR NoKeyErr, WEBERROR NotIntErr);
void TermInit(void);
void TermDeleteAll(void);
int  TermAdd(void);
void TermDelete(int id);
void ErrorForm(EXTENSION_CONTROL_BLOCK *pECB, int
iType, int iErrorNum, int iTermId, int iSyncId, char
*szErrorText, char *szBuffer );
void MakeMainMenuForm(int iTermId, int iSyncId, char
*szForm);
void MakeStockLevelForm(int iTermId, STOCK_LEVEL_DATA
*pStockLevelData, BOOL bInput, char *szForm);
void MakeNewOrderForm(int iTermId, NEW_ORDER_DATA
*pNewOrderData, BOOL bInput, char *szForm);
void MakePaymentForm(int iTermId, PAYMENT_DATA
*pPaymentData, BOOL bInput, char *szForm);
void MakeOrderStatusForm(int iTermId,
ORDER_STATUS_DATA *pOrderStatusData, BOOL bInput,
char *szForm);
```

```
void MakeDeliveryForm(int iTermId, DELIVERY_DATA
*pDeliveryData, BOOL bInput, char *szForm);
void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK
          *pECB, int iTermId, char *szBuffer);
void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, char *szBuffer);
void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, char *szBuffer);
void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, char *szBuffer);
void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK
*pECB, int iTermId, char *szBuffer);
void GetNewOrderData(LPSTR lpszQueryString,
NEW_ORDER_DATA *pNewOrderData);
void GetPaymentData(LPSTR lpszQueryString,
PAYMENT_DATA *pPaymentData);
void GetOrderStatusData(LPSTR lpszQueryString,
ORDER_STATUS_DATA *pOrderStatusData);
BOOL PostDeliveryInfo(long w_id, short o_carrier_id);
BOOL IsNumeric(char *ptr);
BOOL IsDecimal(char *ptr);
void DeliveryWorkerThread(void *ptr);
// Separate function to be able to use Win32
exception handling in
// HttpExtensionProc.
void ProcessCommand(EXTENSION_CONTROL_BLOCK *pECB,
char* szBuffer, int& TermId, int& iSyncId);
```

tpcc_com.cpp

```
/*        FILE:               TPCC_COM.CPP
 *                                    Microsoft
TPC-C Kit Ver. 4.69.000
 *                                    Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                                    not yet
audited
 *
 *        PURPOSE:  Source file for TPC-C COM+ class
implementation.
 *        Contact:  Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *                  4.20.000 - first version
 *                  4.69.000 - updated rev number to
match kit
 */

// needed for CoinitializeEx
#define _WIN32_WINNT 0x0400

#include <windows.h>

// need to declare functions for export
#define DllDecl  __declspec( dllexport )

#include "..\..\common\src\trans.h"
          //tpckit transaction header contains
definations of structures specific to TPC-C
#include "..\..\common\src\error.h"
#include "..\..\common\src\txn_base.h"
#include "..\..\common\src\tpcc_com_errorcode.h"
#include "tpcc_com.h"

#include "..\..\tpcc_com_ps\src\tpcc_com_ps_i.c"
#include "..\..\tpcc_com_all\src\tpcc_com_all_i.c"

// wrapper routine for class constructor
__declspec(dllexport) CTPCC_COM* CTPCC_COM_new(BOOL
bSinglePool)
{
          return new CTPCC_COM(bSinglePool);
}

CTPCC_COM::CTPCC_COM(BOOL bSinglePool)
{
          HRESULT hr = NULL;
          long lRet = 0;
          ULONG ulTmpSize = 0;

          m_pTxn                        = NULL;
          m_pNewOrder                   = NULL;
          m_pPayment                    = NULL;
          m_pStockLevel       = NULL;
          m_pOrderStatus      = NULL;

          m_bSinglePool       = bSinglePool;

          ulTmpSize = (ULONG) sizeof(COM_DATA);
          VariantInit(&m_vTxn);
          m_vTxn.vt = VT_SAFEARRAY;

          m_vTxn.parray =
SafeArrayCreateVector(VT_UI1, ulTmpSize, ulTmpSize);
          if (!m_vTxn.parray)
                    throw new CCOMERR( E_FAIL );

          memset((void*)m_vTxn.parray-
>pvData,0,ulTmpSize);
          m_pTxn = (COM_DATA*)m_vTxn.parray->pvData;

          hr = CoInitializeEx(NULL,
COINIT_MULTITHREADED);
          if (FAILED(hr))
          {
                    throw new CCOMERR( hr );
          }

          // create components
          if (m_bSinglePool)
          {
                    hr = CoCreateInstance(CLSID_TPCC,
NULL, CLSCTX_SERVER, IID_ITPCC, (void
**)&m_pNewOrder);
                    if (FAILED(hr))
                              throw new CCOMERR(hr);

                    // all txns will use same
component
                    m_pPayment = m_pNewOrder;
                    m_pStockLevel = m_pNewOrder;
                    m_pOrderStatus = m_pNewOrder;
          }
          else
          {
                    // use different components for
each txn

                    hr =
CoCreateInstance(CLSID_NewOrder, NULL, CLSCTX_SERVER,
IID_ITPCC, (void **)&m_pNewOrder);
                    if (FAILED(hr))
                              throw new CCOMERR(hr);

                    hr =
CoCreateInstance(CLSID_Payment, NULL, CLSCTX_SERVER,
IID_ITPCC, (void **)&m_pPayment);
                    if (FAILED(hr))
                              throw new CCOMERR(hr);

                    hr =
CoCreateInstance(CLSID_StockLevel, NULL,
CLSCTX_SERVER, IID_ITPCC, (void **)&m_pStockLevel);
                    if (FAILED(hr))
                              throw new CCOMERR(hr);

                    hr =
CoCreateInstance(CLSID_OrderStatus, NULL,
CLSCTX_SERVER, IID_ITPCC, (void **)&m_pOrderStatus);
                    if (FAILED(hr))
                              throw new CCOMERR(hr);
          }

          // call setcomplete to release each
component back into pool
          hr = m_pNewOrder->CallSetComplete();
          if (FAILED(hr))
                    throw new CCOMERR(hr);

          if (!m_bSinglePool)
          {
                    hr = m_pPayment-
>CallSetComplete();
                    if (FAILED(hr))
                              throw new CCOMERR(hr);

                    hr = m_pStockLevel-
>CallSetComplete();
                    if (FAILED(hr))
                              throw new CCOMERR(hr);

                    hr = m_pOrderStatus-
>CallSetComplete();
                    if (FAILED(hr))
                              throw new CCOMERR(hr);
          }
}

CTPCC_COM::~CTPCC_COM()
{
          if (m_pTxn)
                    SafeArrayDestroy(m_vTxn.parray);
```

```
		ReleaseInterface(m_pNewOrder);
		if (!m_bSinglePool)
		{
			ReleaseInterface(m_pPayment);
			ReleaseInterface(m_pStockLevel);
			ReleaseInterface(m_pOrderStatus);
		}
		CoUninitialize();
}

void CTPCC_COM::NewOrder()
{
		VARIANT		vTxn_out;

		HRESULT hr = m_pNewOrder->NewOrder(m_vTxn,
&vTxn_out);

		if (FAILED(hr) && hr != E_TPCCCOM)
			throw new CCOMERR( hr );	//
COM call didn't succeed and there is no output
structure

		memcpy(m_pTxn, (void *)vTxn_out.parray-
>pvData,vTxn_out.parray->rgsabound[0].cElements);
		hr = SafeArrayDestroy(vTxn_out.parray);
		if (hr != S_OK)
			throw new CCOMERR( hr );

		if ( m_pTxn->ErrorType != ERR_SUCCESS )
			throw new CCOMERR( m_pTxn-
>ErrorType, m_pTxn->error );
}

void CTPCC_COM::Payment()
{
		VARIANT		vTxn_out;

		HRESULT hr = m_pPayment->Payment(m_vTxn,
&vTxn_out);

		if (FAILED(hr) && hr != E_TPCCCOM)
			throw new CCOMERR( hr );	//
COM call didn't succeed and there is no output
structure

		memcpy(m_pTxn, (void *)vTxn_out.parray-
>pvData,vTxn_out.parray->rgsabound[0].cElements);
		hr = SafeArrayDestroy(vTxn_out.parray);
		if (hr != S_OK)
			throw new CCOMERR( hr );

		if ( m_pTxn->ErrorType != ERR_SUCCESS )
			throw new CCOMERR( m_pTxn-
>ErrorType, m_pTxn->error );
}

void CTPCC_COM::StockLevel()
{
		VARIANT		vTxn_out;

		HRESULT hr = m_pStockLevel-
>StockLevel(m_vTxn, &vTxn_out);

		if (FAILED(hr) && hr != E_TPCCCOM)
			throw new CCOMERR( hr );	//
COM call didn't succeed and there is no output
structure

		memcpy(m_pTxn, (void *)vTxn_out.parray-
>pvData,vTxn_out.parray->rgsabound[0].cElements);
		hr = SafeArrayDestroy(vTxn_out.parray);
		if (hr != S_OK)
			throw new CCOMERR( hr );

		if ( m_pTxn->ErrorType != ERR_SUCCESS )
			throw new CCOMERR( m_pTxn-
>ErrorType, m_pTxn->error );
}

void CTPCC_COM::OrderStatus()
{
		VARIANT		vTxn_out;

		HRESULT hr = m_pOrderStatus-
>OrderStatus(m_vTxn, &vTxn_out);

		if (FAILED(hr) && hr != E_TPCCCOM)
			throw new CCOMERR( hr );	//
COM call didn't succeed and there is no output
structure

		memcpy(m_pTxn, (void *)vTxn_out.parray-
>pvData,vTxn_out.parray->rgsabound[0].cElements);
		hr = SafeArrayDestroy(vTxn_out.parray);
		if (hr != S_OK)
			throw new CCOMERR( hr );

		if ( m_pTxn->ErrorType != ERR_SUCCESS )
			throw new CCOMERR( m_pTxn-
>ErrorType, m_pTxn->error );
}
```

tpcc_com.h

```
/*	FILE:		TPCC_COM.H
 *					Microsoft
TPC-C Kit Ver. 4.69.000
 *					Copyright
Microsoft, 1999
 *		All Rights Reserved
 *
 *					not yet
audited
 *
 *	PURPOSE:  Header file for TPC-C COM+ class
implementation.
 *
 *  Change history:
 *		4.20.000 - first version
 *		4.69.000 - updated rev number to
match kit
 */

#pragma once

#include <stdio.h>
#include "..\..\tpcc_com_ps\src\tpcc_com_ps.h"

// need to declare functions for import, unless
define has already been created
// by the DLL's .cpp module for export.
#ifndef DllDecl
#define DllDecl  __declspec( dllimport )
#endif

class CCOMERR : public CBaseErr
{
	private:
		char m_szErrorText[64];

	public:
		// use this interface for genuine
COM errors
		CCOMERR( HRESULT hr )
		{
			m_hr = hr;
			m_iErrorType = 0;
			m_iError = 0;
		}

		// use this interface to
impersonate a non-COM error type
		CCOMERR( int iErrorType, int
iError )
		{
			m_iErrorType =
iErrorType;
			m_iError = iError;
			m_hr = S_OK;
		}

		int		m_hr;
		int		m_iErrorType;
		int		m_iError;

		// A CCOMERR class can
impersonate another class, which happens if the error
		// was not actually a COM
Services error, but was simply transmitted back via
COM.
		int ErrorType()
		{
			if (m_iErrorType == 0)
				return
ERR_TYPE_COM;
			else
				return
m_iErrorType;
		}

		char *ErrorTypeStr() { return
"COM"; }

		int ErrorNum()
		{
```

```
                                if (m_iErrorType == 0)
                                        return m_hr;
                        // return COM error
                                else
                                        return
m_iError; // return impersonated error
                        }

                        char *ErrorText()
                        {
                                if (m_hr == S_OK)
                                        sprintf(
m_szErrorText, "Error: Class %d, error # %d",
m_iErrorType, m_iError );
                                else
                                        sprintf(
m_szErrorText, "Error: COM HRESULT %x", m_hr );
                                return m_szErrorText;
                        }
};

class DllDecl CTPCC_COM : public CTPCC_BASE
{
        private:
                BOOL m_bSinglePool;

                // COM Interface pointers
                ITPCC*
        m_pNewOrder;
                ITPCC*
        m_pPayment;
                ITPCC*
        m_pStockLevel;
                ITPCC*
        m_pOrderStatus;

                struct COM_DATA
                {
                        int ErrorType;
                        int error;
                        union
                        {

        NEW_ORDER_DATA                  NewOrder;
                                        PAYMENT_DATA
                Payment;
                                        DELIVERY_DATA
                Delivery;

        STOCK_LEVEL_DATA    StockLevel;

        ORDER_STATUS_DATA   OrderStatus;
                        } u;
                } *m_pTxn;

                VARIANT m_vTxn;
        public:
                CTPCC_COM(BOOL bSinglePool);
                ~CTPCC_COM(void);

                inline PNEW_ORDER_DATA
        BuffAddr_NewOrder()           { return
&m_pTxn->u.NewOrder;    };
                        inline PPAYMENT_DATA
                BuffAddr_Payment()            { return
&m_pTxn->u.Payment;     };
                        inline PDELIVERY_DATA
                BuffAddr_Delivery()           { return
&m_pTxn->u.Delivery;    };
                        inline PSTOCK_LEVEL_DATA
                BuffAddr_StockLevel()         { return
&m_pTxn->u.StockLevel;  };
                        inline PORDER_STATUS_DATA
                BuffAddr_OrderStatus()        { return
&m_pTxn->u.OrderStatus; };

                        void NewOrder                 ();
                        void Payment                  ();
                        void StockLevel               ();
                        void OrderStatus    ();
                        void Delivery                 ()
{ throw new CCOMERR(E_NOTIMPL); }  // not supported

};


inline void ReleaseInterface(IUnknown *pUnk)
{
        if (pUnk)
        {
                pUnk->Release();
                pUnk = NULL;
        }
}


// wrapper routine for class constructor
extern "C" __declspec(dllexport) CTPCC_COM*
CTPCC_COM_new(BOOL);


typedef CTPCC_COM* (TYPE_CTPCC_COM)(BOOL);
```

tpcc_com_all.cpp

```
/*        FILE:               TPCC_COM_ALL.CPP
 *                                      Microsoft
TPC-C Kit Ver. 4.69.000
 *                                      Copyright
Microsoft, 1999
 *             All Rights Reserved
 *
 *                                      Version
4.10.000 audited by Richard Gimarc, Performance
Metrics, 3/17/99
 *
 *        PURPOSE:  Implementation for TPC-C class.
 *        Contact:  Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *                  4.20.000 - updated rev number to
match kit
 *                  4.69.000 - updated rev number to
match kit
 */

#define STRICT
#define _WIN32_WINNT 0x0400
#define _ATL_APARTMENT_THREADED

#include <stdio.h>
#include <atlbase.h>
//You may derive a class from CComModule and use it
if you want to override
//something, but do not change the name of _Module
extern CComModule _Module;

#include <atlcom.h>
#include <initguid.h>
#include <transact.h>
//#include <atlimpl.cpp>
#include <comsvcs.h>

#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>

#include "tpcc_com_ps.h"
#include "..\..\common\src\trans.h"
                                //tpckit transaction
header contains definations of structures specific to
TPC-C
#include "..\..\common\src\txn_base.h"
#include "..\..\common\src\error.h"
#include "..\..\common\src\ReadRegistry.h"
#include "..\..\common\src\tpcc_com_errorcode.h"
#include "..\..\db_odbc_dll\src\tpcc_odbc.h"
          // ODBC implementation of TPC-C txns

#include "resource.h"
#include "tpcc_com_all.h"
#include "tpcc_com_all_i.c"
#include "Methods.h"
#include "..\..\tpcc_com_ps\src\tpcc_com_ps_i.c"
#include "..\..\common\src\ReadRegistry.cpp"

CComModule _Module;

BEGIN_OBJECT_MAP(ObjectMap)
          OBJECT_ENTRY(CLSID_TPCC, CTPCC)
          OBJECT_ENTRY(CLSID_NewOrder, CNewOrder)
          OBJECT_ENTRY(CLSID_OrderStatus,
COrderStatus)
          OBJECT_ENTRY(CLSID_Payment, CPayment)
          OBJECT_ENTRY(CLSID_StockLevel, CStockLevel)
END_OBJECT_MAP()


// configuration settings from registry
TPCCREGISTRYDATA    Reg;
char
          szMyComputerName[MAX_COMPUTERNAME_LENGTH+1]
;
```

```
static  HINSTANCE hLibInstanceDb = NULL;

TYPE_CTPCC_ODBC               *pCTPCC_ODBC_new;

// Critical section to synchronize connection open
and close.
//
CRITICAL_SECTION hConnectCriticalSection;

////////////////////////////////////////////////////
//////////////////////
// DLL Entry Point

extern "C"
BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD
dwReason, LPVOID /*lpReserved*/)
{
          char szDllName[128];

          try
          {
                    if (dwReason ==
DLL_PROCESS_ATTACH)
                    {
                              _Module.Init(ObjectMap,
hInstance);

          DisableThreadLibraryCalls(hInstance);

                              DWORD dwSize =
MAX_COMPUTERNAME_LENGTH+1;

          GetComputerName(szMyComputerName, &dwSize);

          szMyComputerName[dwSize] = 0;

                              if (
ReadTPCCRegistrySettings( &Reg ) )
                                        throw new
CCOMPONENT_ERR( ERR_MISSING_REGISTRY_ENTRIES );

                              if (Reg.eDB_Protocol ==
ODBC)
                              {
                                        strcpy(
szDllName, Reg.szPath );
                                        strcat(
szDllName, "tpcc_odbc.dll");

          hLibInstanceDb = LoadLibrary( szDllName );
                                        if
(hLibInstanceDb == NULL)

          throw new CCOMPONENT_ERR(
ERR_LOADDLL_FAILED, szDllName, GetLastError() );

                                        // get
function pointer to wrapper for class constructor

          pCTPCC_ODBC_new = (TYPE_CTPCC_ODBC*)
GetProcAddress(hLibInstanceDb,"CTPCC_ODBC_new");
                                        if
(pCTPCC_ODBC_new == NULL)

          throw new CCOMPONENT_ERR(
ERR_GETPROCADDR_FAILED, szDllName, GetLastError() );
                                        }
                                        else
                                                  throw new
CCOMPONENT_ERR( ERR_UNKNOWN_DB_PROTOCOL );

                                        if (Reg.dwConnectDelay
> 0)
                                        {

          InitializeCriticalSection(&hConnectCritical
Section);
                                        }

                    }
                    else if (dwReason ==
DLL_PROCESS_DETACH)
                              _Module.Term();

          }
          catch (CBaseErr *e)
          {
                    TCHAR szMsg[256];

                    _sntprintf(szMsg, sizeof(szMsg),
"%s error, code %d: %s",
                                                  e-
>ErrorTypeStr(), e->ErrorNum(), e->ErrorText());
                    WriteMessageToEventLog( szMsg );

                    delete e;
                    return FALSE;
          }
          catch (...)
          {

          WriteMessageToEventLog(TEXT("Unhandled
exception in object DllMain"));
                    return FALSE;
          }

          return TRUE;        // OK
}

////////////////////////////////////////////////////
//////////////////////
// Used to determine whether the DLL can be unloaded
by OLE

STDAPI DllCanUnloadNow(void)
{
          return (_Module.GetLockCount()==0) ? S_OK :
S_FALSE;
}

////////////////////////////////////////////////////
//////////////////////
// Returns a class factory to create an object of the
requested type

STDAPI DllGetClassObject(REFCLSID rclsid, REFIID
riid, LPVOID* ppv)
{
          return _Module.GetClassObject(rclsid, riid,
ppv);
}

////////////////////////////////////////////////////
//////////////////////
// DllRegisterServer - Adds entries to the system
registry

STDAPI DllRegisterServer(void)
{
          // registers object, typelib and all
interfaces in typelib
          return _Module.RegisterServer(TRUE);
}

////////////////////////////////////////////////////
//////////////////////
// DllUnregisterServer - Removes entries from the
system registry

STDAPI DllUnregisterServer(void)
{
          _Module.UnregisterServer();
          return S_OK;
}


static void WriteMessageToEventLog(LPTSTR lpszMsg)
{
    TCHAR   szMsg[256];
    HANDLE  hEventSource;
    LPTSTR  lpszStrings[2];

    // Use event logging to log the error.
    //
    hEventSource = RegisterEventSource(NULL,
TEXT("tpcc_com_all.dll"));

    _stprintf(szMsg, TEXT("Error in COM+ TPC-C
Component: "));
    lpszStrings[0] = szMsg;
    lpszStrings[1] = lpszMsg;

    if (hEventSource != NULL)
          {
        ReportEvent(hEventSource, // handle of event
source
            EVENTLOG_ERROR_TYPE,  // event type
            0,                    // event category
            0,                    // event ID
            NULL,                 // current user's
SID
            2,                    // strings in
lpszStrings
            0,                    // no bytes of raw
data
            (LPCTSTR *)lpszStrings,    // array of
error strings
```

```
            NULL);               // no raw data

        (VOID) DeregisterEventSource(hEventSource);
    }
}

inline void ReleaseInterface(IUnknown *pUnk)
{
          if (pUnk)
          {
                    pUnk->Release();
                    pUnk = NULL;
          }
}

/* FUNCTION: CCOMPONENT_ERR::ErrorText
 *
 */

char* CCOMPONENT_ERR::ErrorText(void)
{
          static SERRORMSG errorMsgs[] =
          {
                    { ERR_MISSING_REGISTRY_ENTRIES,
          "Required entries missing from registry."
                                        },
                    { ERR_LOADDLL_FAILED,
                    "Load of DLL failed.  DLL="

                    },
                    { ERR_GETPROCADDR_FAILED,
          "Could not map proc in DLL.  GetProcAddr
error.  DLL="       },
                    { ERR_UNKNOWN_DB_PROTOCOL,
          "Unknown database protocol specified in
registry."                    },
                    { 0,                        ""



                              }
          };

          char szTmp[256];
          int i = 0;
          while (TRUE)
          {
                    if (errorMsgs[i].szMsg[0] == 0)
                    {
                              strcpy( szTmp, "Unknown
error number." );
                              break;
                    }
                    if (m_Error ==
errorMsgs[i].iError)
                    {
                              strcpy( szTmp,
errorMsgs[i].szMsg );
                              break;
                    }
                    i++;
          }
          if (m_szTextDetail)
                    strcat( szTmp, m_szTextDetail );
          if (m_SystemErr)
                    wsprintf( szTmp+strlen(szTmp), "
Error=%d", m_SystemErr );

          m_szErrorText = new char[strlen(szTmp)+1];
          strcpy( m_szErrorText, szTmp );
          return m_szErrorText;
}


CTPCC_Common::CTPCC_Common()
{
          m_pTxn = NULL;
          m_bCanBePooled = TRUE;
}

CTPCC_Common::~CTPCC_Common()
{
          // Pace connection close for VIA.
          //
          if (Reg.dwConnectDelay > 0)
          {

          EnterCriticalSection(&hConnectCriticalSecti
on);

                    Sleep(Reg.dwConnectDelay);


          LeaveCriticalSection(&hConnectCriticalSecti
on);
          }

          if (m_pTxn)
          {
                    delete m_pTxn;
          }
}

HRESULT CTPCC_Common::CallSetComplete()
{
          IObjectContext* pObjectContext = NULL;

          // get our object context
          HRESULT hr = CoGetObjectContext(
IID_IObjectContext, (void **)&pObjectContext );
          pObjectContext->SetComplete();
          ReleaseInterface(pObjectContext);
          return hr;
}

//
//        called by the ctor activator
//
STDMETHODIMP CTPCC_Common::Construct(IDispatch *
pUnk)
{
          // Code to access construction string, if
needed later...
          //        if (!pUnk)
          //                  return E_UNEXPECTED;
          //        IObjectConstructString * pString
= NULL;
          //        HRESULT hr = pUnk-
>QueryInterface(IID_IObjectConstructString, (void
**)&pString);
          //        pString->Release();

          try
          {
                    // Pace connection creation for
VIA.
                    //
                    if (Reg.dwConnectDelay > 0)
                    {

          EnterCriticalSection(&hConnectCriticalSecti
on);


                    Sleep(Reg.dwConnectDelay);


          LeaveCriticalSection(&hConnectCriticalSecti
on);
                    }

                    if (Reg.eDB_Protocol == ODBC)
                              m_pTxn =
pCTPCC_ODBC_new(    Reg.szDbServer, Reg.szDbUser,
Reg.szDbPassword,


          szMyComputerName, Reg.szDbName,


          Reg.szSPPrefix,
Reg.bCallNoDuplicatesNewOrder );
          }
          catch (CBaseErr *e)
          {
                    TCHAR szMsg[256];

                    _sntprintf(szMsg, sizeof(szMsg),
"%s error in CTPCC_Common::Construct, code %d: %s",
                                                  e-
>ErrorTypeStr(), e->ErrorNum(), e->ErrorText());
                    WriteMessageToEventLog( szMsg );
                    delete e;
                    return E_FAIL;
          }
          catch (...)
          {

          WriteMessageToEventLog(TEXT("Unhandled
exception in object ::Construct"));
                    return E_FAIL;
          }

          return S_OK;
}

HRESULT CTPCC_Common::NewOrder(VARIANT txn_in,
VARIANT* txn_out)
```

```
{
        PNEW_ORDER_DATA     pNewOrder;
        COM_DATA            *pData;
        COM_DATA            *pOutData;

        try
        {
                // Allocate output structure
first because it is also used in the catch clauses.
                //
                VariantInit(txn_out);
                txn_out->vt = VT_SAFEARRAY;
                txn_out->parray =
SafeArrayCreateVector( VT_UI1,

                txn_in.parray->rgsabound-
>cElements,

                txn_in.parray->rgsabound-
>cElements);
                if (txn_out->parray == NULL)  //
sanity error checking - for very rare case, but to be
sure
                {
                        return E_OUTOFMEMORY;
                }

                pOutData = (COM_DATA*)txn_out-
>parray->pvData;

                pData = (COM_DATA*)txn_in.parray-
>pvData;
                pNewOrder = m_pTxn-
>BuffAddr_NewOrder();

                memcpy(pNewOrder, &pData-
>u.NewOrder, sizeof(NEW_ORDER_DATA));

                m_pTxn->NewOrder();           //
do the actual txn

                memcpy( &pOutData->u.NewOrder,
pNewOrder, sizeof(NEW_ORDER_DATA));

                pOutData->retval = ERR_SUCCESS;
                pOutData->error = 0;
                return S_OK;
        }
        catch (CBaseErr *e)
        {
                // check for lost database
connection; if yes, component is toast
                if ( ((e->ErrorType() ==
ERR_TYPE_ODBC) && (e->ErrorNum() == 10054)) )
                        m_bCanBePooled = FALSE;

                pOutData->retval = e-
>ErrorType();
                pOutData->error = e->ErrorNum();
                delete e;
                return E_TPCCCOM;
        }
        catch (...)
        {

        WriteMessageToEventLog(TEXT("Unhandled
exception in CTPCC_Common::NewOrder."));
                pOutData->retval =
ERR_TYPE_LOGIC;
                pOutData->error = 0;
                m_bCanBePooled = FALSE;
                return E_TPCCCOM;
        }
}

HRESULT CTPCC_Common::Payment(VARIANT txn_in,
VARIANT* txn_out)
{
        PPAYMENT_DATA       pPayment;
        COM_DATA            *pData;
        COM_DATA            *pOutData;

        try
        {
                // Allocate output structure
first because it is also used in the catch clauses.
                //
                VariantInit(txn_out);
                txn_out->vt = VT_SAFEARRAY;
                txn_out->parray =
SafeArrayCreateVector( VT_UI1,

                txn_in.parray->rgsabound-
>cElements,

                txn_in.parray->rgsabound-
>cElements);
                if (txn_out->parray == NULL)  //
sanity error checking - for very rare case, but to be
sure
                {
                        return E_OUTOFMEMORY;
                }

                pOutData = (COM_DATA*)txn_out-
>parray->pvData;

                pData = (COM_DATA*)txn_in.parray-
>pvData;
                pPayment = m_pTxn-
>BuffAddr_Payment();

                memcpy(pPayment, &pData-
>u.Payment, sizeof(PAYMENT_DATA));

                m_pTxn->Payment();            //
do the actual txn

                memcpy( &pOutData->u.Payment,
pPayment, sizeof(PAYMENT_DATA));

                pOutData->retval = ERR_SUCCESS;
                pOutData->error = 0;
                return S_OK;
        }
        catch (CBaseErr *e)
        {
                // check for lost database
connection; if yes, component is toast
                if ( ((e->ErrorType() ==
ERR_TYPE_ODBC) && (e->ErrorNum() == 10054)) )
                        m_bCanBePooled = FALSE;

                pOutData->retval = e-
>ErrorType();
                pOutData->error = e->ErrorNum();
                delete e;
                return E_TPCCCOM;
        }
        catch (...)
        {

        WriteMessageToEventLog(TEXT("Unhandled
exception in CTPCC_Common::Payment."));
                pOutData->retval =
ERR_TYPE_LOGIC;
                pOutData->error = 0;
                m_bCanBePooled = FALSE;
                return E_TPCCCOM;
        }
}

HRESULT CTPCC_Common::StockLevel(VARIANT txn_in,
VARIANT* txn_out)
{
        PSTOCK_LEVEL_DATA   pStockLevel;
        COM_DATA                    *pData;
        COM_DATA                    *pOutData;

        try
        {
                // Allocate output structure
first because it is also used in the catch clauses.
                //
                VariantInit(txn_out);
                txn_out->vt = VT_SAFEARRAY;
                txn_out->parray =
SafeArrayCreateVector( VT_UI1,

                txn_in.parray->rgsabound-
>cElements,

                txn_in.parray->rgsabound-
>cElements);
                if (txn_out->parray == NULL)  //
sanity error checking - for very rare case, but to be
sure
                {
                        return E_OUTOFMEMORY;
                }

                pOutData = (COM_DATA*)txn_out-
>parray->pvData;

                pData = (COM_DATA*)txn_in.parray-
>pvData;
                pStockLevel = m_pTxn-
>BuffAddr_StockLevel();
```

```
                    memcpy(pStockLevel, &pData-
>u.StockLevel, sizeof(STOCK_LEVEL_DATA));

                    m_pTxn->StockLevel();

                    memcpy( &pOutData->u.StockLevel,
pStockLevel, sizeof(STOCK_LEVEL_DATA));

                    pOutData->retval = ERR_SUCCESS;
                    pOutData->error = 0;
                    return S_OK;
          }
          catch (CBaseErr *e)
          {
                    // check for lost database
connection; if yes, component is toast
                    if ( ((e->ErrorType() ==
ERR_TYPE_ODBC) && (e->ErrorNum() == 10054)) )
                              m_bCanBePooled = FALSE;

                    pOutData->retval = e-
>ErrorType();
                    pOutData->error = e->ErrorNum();
                    delete e;
                    return E_TPCCCOM;
          }
          catch (...)
          {

          WriteMessageToEventLog(TEXT("Unhandled
exception in CTPCC_Common::StockLevel."));
                    pOutData->retval =
ERR_TYPE_LOGIC;
                    pOutData->error = 0;
                    m_bCanBePooled = FALSE;
                    return E_TPCCCOM;
          }
}

HRESULT CTPCC_Common::OrderStatus(VARIANT txn_in,
VARIANT* txn_out)
{
          PORDER_STATUS_DATA  pOrderStatus;
          COM_DATA                    *pData;
          COM_DATA                    *pOutData;
          try
          {
                    // Allocate output structure
first because it is also used in the catch clauses.
                    //
                    VariantInit(txn_out);
                    txn_out->vt = VT_SAFEARRAY;
                    txn_out->parray =
SafeArrayCreateVector( VT_UI1,

                    txn_in.parray->rgsabound-
>cElements,

                    txn_in.parray->rgsabound-
>cElements);
                    if (txn_out->parray == NULL)  //
sanity error checking - for very rare case, but to be
sure
                    {
                              return E_OUTOFMEMORY;
                    }

                    pOutData = (COM_DATA*)txn_out-
>parray->pvData;

                    pData = (COM_DATA*)txn_in.parray-
>pvData;
                    pOrderStatus = m_pTxn-
>BuffAddr_OrderStatus();

                    memcpy(pOrderStatus, &pData-
>u.OrderStatus, sizeof(ORDER_STATUS_DATA));

                    m_pTxn->OrderStatus();

                    memcpy( &pOutData->u.OrderStatus,
pOrderStatus, sizeof(ORDER_STATUS_DATA));

                    pOutData->retval = ERR_SUCCESS;
                    pOutData->error = 0;
                    return S_OK;
          }
          catch (CBaseErr *e)
          {
                    // check for lost database
connection; if yes, component is toast
                    if ( ((e->ErrorType() ==
ERR_TYPE_ODBC) && (e->ErrorNum() == 10054)) )
                              m_bCanBePooled = FALSE;

                    pOutData->retval = e-
>ErrorType();
                    pOutData->error = e->ErrorNum();
                    delete e;
                    return E_TPCCCOM;
          }
          catch (...)
          {

          WriteMessageToEventLog(TEXT("Unhandled
exception in CTPCC_Common::OrderStatus."));
                    pOutData->retval =
ERR_TYPE_LOGIC;
                    pOutData->error = 0;
                    m_bCanBePooled = FALSE;
                    return E_TPCCCOM;
          }
}
```

tpcc_com_all.def

```
; tpcc_com_all.def : Declares the module parameters.

LIBRARY      "tpcc_com_all.dll"

EXPORTS
          DllCanUnloadNow     PRIVATE
          DllGetClassObject   PRIVATE
          DllRegisterServer   PRIVATE
          DllUnregisterServer PRIVATE
```

tpcc_com_all.h

```
/* this ALWAYS GENERATED file contains the
definitions for the interfaces */


 /* File created by MIDL compiler version 6.00.0361
*/
/* at Tue Nov 10 10:51:22 2009
 */
/* Compiler settings for .\src\tpcc_com_all.idl:
    Oicf, W1, Zp8, env=Win32 (32b run)
    protocol : dce , ms_ext, c_ext, robust
    error checks: allocation ref bounds_check enum
stub_data
    VC __declspec() decoration level:
         __declspec(uuid()), __declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(), MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#pragma warning( disable: 4049 )  /* more than 64k
source lines */


/* verify that the <rpcndr.h> version is high enough
to compile this file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 475
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __RPCNDR_H_VERSION__
#error this stub requires an updated version of
<rpcndr.h>
#endif // __RPCNDR_H_VERSION__


#ifndef __tpcc_com_all_h__
#define __tpcc_com_all_h__

#if defined(_MSC_VER) && (_MSC_VER >= 1020)
#pragma once
#endif

/* Forward Declarations */

#ifndef __TPCC_FWD_DEFINED__
#define __TPCC_FWD_DEFINED__

#ifdef __cplusplus
typedef class TPCC TPCC;
```

```
#else
typedef struct TPCC TPCC;
#endif /* __cplusplus */

#endif    /* __TPCC_FWD_DEFINED__ */


#ifndef __NewOrder_FWD_DEFINED__
#define __NewOrder_FWD_DEFINED__

#ifdef __cplusplus
typedef class NewOrder NewOrder;
#else
typedef struct NewOrder NewOrder;
#endif /* __cplusplus */

#endif    /* __NewOrder_FWD_DEFINED__ */


#ifndef __OrderStatus_FWD_DEFINED__
#define __OrderStatus_FWD_DEFINED__

#ifdef __cplusplus
typedef class OrderStatus OrderStatus;
#else
typedef struct OrderStatus OrderStatus;
#endif /* __cplusplus */

#endif    /* __OrderStatus_FWD_DEFINED__ */


#ifndef __Payment_FWD_DEFINED__
#define __Payment_FWD_DEFINED__

#ifdef __cplusplus
typedef class Payment Payment;
#else
typedef struct Payment Payment;
#endif /* __cplusplus */

#endif    /* __Payment_FWD_DEFINED__ */


#ifndef __StockLevel_FWD_DEFINED__
#define __StockLevel_FWD_DEFINED__

#ifdef __cplusplus
typedef class StockLevel StockLevel;
#else
typedef struct StockLevel StockLevel;
#endif /* __cplusplus */

#endif    /* __StockLevel_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"
#include "tpcc_com_ps.h"

#ifdef __cplusplus
extern "C"{
#endif

void * __RPC_USER MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void * );

/* interface __MIDL_itf_tpcc_com_all_0000 */
/* [local] */




extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_all_0000_v0_0_c_ifspec;
extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_all_0000_v0_0_s_ifspec;


#ifndef __TPCCLib_LIBRARY_DEFINED__
#define __TPCCLib_LIBRARY_DEFINED__

/* library TPCCLib */
/* [helpstring][version][uuid] */


EXTERN_C const IID LIBID_TPCCLib;

EXTERN_C const CLSID CLSID_TPCC;

#ifdef __cplusplus

class DECLSPEC_UUID("122A3128-2520-11D3-BA71-
00C04FBFE08B")
TPCC;
#endif

EXTERN_C const CLSID CLSID_NewOrder;

#ifdef __cplusplus

class DECLSPEC_UUID("975BAABF-84A7-11D2-BA47-
00C04FBFE08B")
NewOrder;
#endif

EXTERN_C const CLSID CLSID_OrderStatus;

#ifdef __cplusplus

class DECLSPEC_UUID("266836AD-A50D-11D2-BA4E-
00C04FBFE08B")
OrderStatus;
#endif

EXTERN_C const CLSID CLSID_Payment;

#ifdef __cplusplus

class DECLSPEC_UUID("CD02F7EF-A4FA-11D2-BA4E-
00C04FBFE08B")
Payment;
#endif

EXTERN_C const CLSID CLSID_StockLevel;

#ifdef __cplusplus

class DECLSPEC_UUID("2668369E-A50D-11D2-BA4E-
00C04FBFE08B")
StockLevel;
#endif
#endif /* __TPCCLib_LIBRARY_DEFINED__ */

/* Additional Prototypes for ALL interfaces */

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

tpcc_com_all.idl

```
/*        FILE:               TPCC.IDL
 *                                          Microsoft
TPC-C Kit Ver. 4.69.000
 *                                          Copyright
Microsoft, 1999
 *             All Rights Reserved
 *
 *                                          not yet
audited
 *
 *        PURPOSE:  IDL source for TPCC.dll.  This
file is processed by the MIDL tool to
 *                                          produce the
type library (TPCC.tlb) and marshalling code.
 *
 *  Change history:
 *                  4.20.000 - first version
 *                  4.69.000 - updated rev number to
match kit
 */

interface TPCC;
interface NewOrder;
interface OrderStatus;
interface Payment;
interface StockLevel;


import "oaidl.idl";
import "ocidl.idl";
import "..\tpcc_com_ps\src\tpcc_com_ps.idl";

[
          uuid(122A3117-2520-11D3-BA71-00C04FBFE08B),
```

```
        version(1.0),
        helpstring("TPC-C 1.0 Type Library")
]
library TPCCLib
{
        importlib("stdole32.tlb");
        importlib("stdole2.tlb");

        [
                uuid(122A3128-2520-11D3-BA71-
00C04FBFE08B),
                helpstring("All Txns Class")
        ]
        coclass TPCC
        {
                [default] interface ITPCC;
        };


        [
                uuid(975BAABF-84A7-11D2-BA47-
00C04FBFE08B),
                helpstring("NewOrder Class")
        ]
        coclass NewOrder
        {
                [default] interface ITPCC;
        };


        [
                uuid(266836AD-A50D-11D2-BA4E-
00C04FBFE08B),
                helpstring("OrderStatus Class")
        ]
        coclass OrderStatus
        {
                [default] interface ITPCC;
        };


        [
                uuid(CD02F7EF-A4FA-11D2-BA4E-
00C04FBFE08B),
                helpstring("Payment Class")
        ]
        coclass Payment
        {
                [default] interface ITPCC;
        };


        [
                uuid(2668369E-A50D-11D2-BA4E-
00C04FBFE08B),
                helpstring("StockLevel Class")
        ]
        coclass StockLevel
        {
                [default] interface ITPCC;
        };

};
```

tpcc_com_all_i.c

```
/* this ALWAYS GENERATED file contains the IIDs and
CLSIDs */

/* link this file in with the server and any clients
*/


 /* File created by MIDL compiler version 6.00.0361
*/
/* at Tue Nov 10 10:51:22 2009
 */
/* Compiler settings for .\src\tpcc_com_all.idl:
    Oicf, W1, Zp8, env=Win32 (32b run)
    protocol : dce , ms_ext, c_ext, robust
    error checks: allocation ref bounds_check enum
stub_data
    VC __declspec() decoration level:
         __declspec(uuid()), __declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(), MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if !defined(_M_IA64) && !defined(_M_AMD64)


#pragma warning( disable: 4049 )  /* more than 64k
source lines */


#ifdef __cplusplus
extern "C"{
#endif


#include <rpc.h>
#include <rpcndr.h>

#ifdef _MIDL_USE_GUIDDEF_

#ifndef INITGUID
#define INITGUID
#include <guiddef.h>
#undef INITGUID
#else
#include <guiddef.h>
#endif

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,
b7,b8) \

DEFINE_GUID(name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8)

#else // !_MIDL_USE_GUIDDEF_

#ifndef __IID_DEFINED__

#define __IID_DEFINED__

typedef struct _IID
{
    unsigned long x;
    unsigned short s1;
    unsigned short s2;
    unsigned char  c[8];
} IID;

#endif // __IID_DEFINED__

#ifndef CLSID_DEFINED
#define CLSID_DEFINED
typedef IID CLSID;
#endif // CLSID_DEFINED

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,
b7,b8) \
        const type name =
{l,w1,w2,{b1,b2,b3,b4,b5,b6,b7,b8}}

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
LIBID_TPCCLib,0x122A3117,0x2520,0x11D3,0xBA,0x71,0x00
,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_TPCC,0x122A3128,0x2520,0x11D3,0xBA,0x71,0x00,0x
C0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_NewOrder,0x975BAABF,0x84A7,0x11D2,0xBA,0x47,0x0
0,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_OrderStatus,0x266836AD,0xA50D,0x11D2,0xBA,0x4E,
0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_Payment,0xCD02F7EF,0xA4FA,0x11D2,0xBA,0x4E,0x00
,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_StockLevel,0x2668369E,0xA50D,0x11D2,0xBA,0x4E,0
x00,0xC0,0x4F,0xBF,0xE0,0x8B);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif



#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/
```

```
/* this ALWAYS GENERATED file contains the IIDs and
CLSIDs */

/* link this file in with the server and any clients
*/


 /* File created by MIDL compiler version 6.00.0361
*/
/* at Tue Nov 10 10:51:22 2009
 */
/* Compiler settings for .\src\tpcc_com_all.idl:
    Oicf, W1, Zp8, env=Win64 (32b run,appending)
    protocol : dce , ms_ext, c_ext, robust
    error checks: allocation ref bounds_check enum
stub_data
    VC __declspec() decoration level:
         __declspec(uuid()), __declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(), MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if defined(_M_IA64) || defined(_M_AMD64)


#pragma warning( disable: 4049 )  /* more than 64k
source lines */


#ifdef __cplusplus
extern "C"{
#endif


#include <rpc.h>
#include <rpcndr.h>

#ifdef _MIDL_USE_GUIDDEF_

#ifndef INITGUID
#define INITGUID
#include <guiddef.h>
#undef INITGUID
#else
#include <guiddef.h>
#endif

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,
b7,b8) \

DEFINE_GUID(name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8)

#else // !_MIDL_USE_GUIDDEF_

#ifndef __IID_DEFINED__
#define __IID_DEFINED__

typedef struct _IID
{
    unsigned long x;
    unsigned short s1;
    unsigned short s2;
    unsigned char  c[8];
} IID;

#endif // __IID_DEFINED__

#ifndef CLSID_DEFINED
#define CLSID_DEFINED
typedef IID CLSID;
#endif // CLSID_DEFINED

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,
b7,b8) \
        const type name =
{l,w1,w2,{b1,b2,b3,b4,b5,b6,b7,b8}}

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
LIBID_TPCCLib,0x122A3117,0x2520,0x11D3,0xBA,0x71,0x00
,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_TPCC,0x122A3128,0x2520,0x11D3,0xBA,0x71,0x00,0x
C0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_NewOrder,0x975BAABF,0x84A7,0x11D2,0xBA,0x47,0x0
0,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_OrderStatus,0x266836AD,0xA50D,0x11D2,0xBA,0x4E,
0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_Payment,0xCD02F7EF,0xA4FA,0x11D2,0xBA,0x4E,0x00
,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_StockLevel,0x2668369E,0xA50D,0x11D2,0xBA,0x4E,0
x00,0xC0,0x4F,0xBF,0xE0,0x8B);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif



#endif /* defined(_M_IA64) || defined(_M_AMD64)*/
```

tpcc_com_errorcode.h

```
/*        FILE:                TPCC_COM_ERRORCODE.H
 *                                          Microsoft
TPC-C Kit Ver. 4.20.000
 *                                          Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                                          not yet
audited
 *
 *        PURPOSE:  Header file defining the error
code returned from ITPCC COM interface.
 *
 *  Change history:
 *                  4.20.000 - first version
 */

// Error return value for methods in ITPCC interface.
//
// Define as 0x80042345 (decimal -2147212475 ).
//
const HRESULT E_TPCCCOM = MAKE_HRESULT
(SEVERITY_ERROR, FACILITY_ITF, 0x2345);
```

tpcc_com_ps.def

```
LIBRARY      "tpcc_com_ps"

EXPORTS
          DllGetClassObject       PRIVATE
          DllCanUnloadNow         PRIVATE
          GetProxyDllInfo         PRIVATE
          DllRegisterServer             PRIVATE
          DllUnregisterServer           PRIVATE
```

tpcc_com_ps.h

```
/* this ALWAYS GENERATED file contains the
definitions for the interfaces */


 /* File created by MIDL compiler version 6.00.0361
*/
/* at Tue Nov 10 10:51:13 2009
 */
/* Compiler settings for .\src\tpcc_com_ps.idl:
    Oicf, W1, Zp8, env=Win32 (32b run)
    protocol : dce , ms_ext, c_ext
    error checks: allocation ref bounds_check enum
stub_data
    VC __declspec() decoration level:
```

```
        __declspec(uuid()), __declspec(selectany),
__declspec(novtable)
        DECLSPEC_UUID(), MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#pragma warning( disable: 4049 )  /* more than 64k
source lines */


/* verify that the <rpcndr.h> version is high enough
to compile this file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 440
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __RPCNDR_H_VERSION__
#error this stub requires an updated version of
<rpcndr.h>
#endif // __RPCNDR_H_VERSION__

#ifndef COM_NO_WINDOWS_H
#include "windows.h"
#include "ole2.h"
#endif /*COM_NO_WINDOWS_H*/

#ifndef __tpcc_com_ps_h__
#define __tpcc_com_ps_h__

#if defined(_MSC_VER) && (_MSC_VER >= 1020)
#pragma once
#endif

/* Forward Declarations */

#ifndef __ITPCC_FWD_DEFINED__
#define __ITPCC_FWD_DEFINED__
typedef interface ITPCC ITPCC;
#endif    /* __ITPCC_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"

#ifdef __cplusplus
extern "C"{
#endif

void * __RPC_USER MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void * );

/* interface __MIDL_itf_tpcc_com_ps_0000 */
/* [local] */




extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_ps_0000_v0_0_c_ifspec;
```

```
extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_ps_0000_v0_0_s_ifspec;

#ifndef __ITPCC_INTERFACE_DEFINED__
#define __ITPCC_INTERFACE_DEFINED__

/* interface ITPCC */
/* [unique][helpstring][uuid][oleautomation][object]
*/


EXTERN_C const IID IID_ITPCC;

#if defined(__cplusplus) && !defined(CINTERFACE)

    MIDL_INTERFACE("FEEE6AA2-84B1-11d2-BA47-
00C04FBFE08B")
    ITPCC : public IUnknown
    {
    public:
        virtual HRESULT __stdcall NewOrder(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out) = 0;

        virtual HRESULT __stdcall Payment(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out) = 0;

        virtual HRESULT __stdcall Delivery(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out) = 0;

        virtual HRESULT __stdcall StockLevel(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out) = 0;

        virtual HRESULT __stdcall OrderStatus(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out) = 0;

        virtual HRESULT __stdcall CallSetComplete(
void) = 0;

    };

#else     /* C style interface */

    typedef struct ITPCCVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE *QueryInterface
)(
            ITPCC * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void **ppvObject);

        ULONG ( STDMETHODCALLTYPE *AddRef )(
            ITPCC * This);

        ULONG ( STDMETHODCALLTYPE *Release )(
            ITPCC * This);
```

```
        HRESULT ( __stdcall *NewOrder )(
            ITPCC * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out);

        HRESULT ( __stdcall *Payment )(
            ITPCC * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out);

        HRESULT ( __stdcall *Delivery )(
            ITPCC * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out);

        HRESULT ( __stdcall *StockLevel )(
            ITPCC * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out);

        HRESULT ( __stdcall *OrderStatus )(
            ITPCC * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT *txn_out);

        HRESULT ( __stdcall *CallSetComplete )(
            ITPCC * This);

        END_INTERFACE
    } ITPCCVtbl;

    interface ITPCC
    {
        CONST_VTBL struct ITPCCVtbl *lpVtbl;
    };



#ifdef COBJMACROS


#define ITPCC_QueryInterface(This,riid,ppvObject) \
    (This)->lpVtbl ->
QueryInterface(This,riid,ppvObject)

#define ITPCC_AddRef(This)    \
    (This)->lpVtbl -> AddRef(This)

#define ITPCC_Release(This)   \
    (This)->lpVtbl -> Release(This)


#define ITPCC_NewOrder(This,txn_in,txn_out)       \
    (This)->lpVtbl -> NewOrder(This,txn_in,txn_out)

#define ITPCC_Payment(This,txn_in,txn_out)        \
    (This)->lpVtbl -> Payment(This,txn_in,txn_out)

#define ITPCC_Delivery(This,txn_in,txn_out)       \
    (This)->lpVtbl -> Delivery(This,txn_in,txn_out)

#define ITPCC_StockLevel(This,txn_in,txn_out)     \
    (This)->lpVtbl -> StockLevel(This,txn_in,txn_out)
```

```
#define ITPCC_OrderStatus(This,txn_in,txn_out)    \
    (This)->lpVtbl ->
OrderStatus(This,txn_in,txn_out)

#define ITPCC_CallSetComplete(This)     \
    (This)->lpVtbl -> CallSetComplete(This)

#endif /* COBJMACROS */


#endif    /* C style interface */



HRESULT __stdcall ITPCC_NewOrder_Proxy(
    ITPCC * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT *txn_out);


void __RPC_STUB ITPCC_NewOrder_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


HRESULT __stdcall ITPCC_Payment_Proxy(
    ITPCC * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT *txn_out);


void __RPC_STUB ITPCC_Payment_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


HRESULT __stdcall ITPCC_Delivery_Proxy(
    ITPCC * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT *txn_out);


void __RPC_STUB ITPCC_Delivery_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


HRESULT __stdcall ITPCC_StockLevel_Proxy(
    ITPCC * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT *txn_out);


void __RPC_STUB ITPCC_StockLevel_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


HRESULT __stdcall ITPCC_OrderStatus_Proxy(
    ITPCC * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT *txn_out);


void __RPC_STUB ITPCC_OrderStatus_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


HRESULT __stdcall ITPCC_CallSetComplete_Proxy(
    ITPCC * This);


void __RPC_STUB ITPCC_CallSetComplete_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);



#endif    /* __ITPCC_INTERFACE_DEFINED__ */


/* Additional Prototypes for ALL interfaces */

unsigned long             __RPC_USER
VARIANT_UserSize(     unsigned long *, unsigned long
, VARIANT * );
unsigned char * __RPC_USER  VARIANT_UserMarshal(
unsigned long *, unsigned char *, VARIANT * );
unsigned char * __RPC_USER
VARIANT_UserUnmarshal(unsigned long *, unsigned char
*, VARIANT * );
void                      __RPC_USER
VARIANT_UserFree(     unsigned long *, VARIANT * );

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

tpcc_com_ps.idl

```
/*        FILE:              ITPCC.IDL
 *                                        Microsoft
TPC-C Kit Ver. 4.20.000
 *                                        Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                                        not yet
audited
 *
 *        PURPOSE:  Defines the interface used by
TPCC. This interface can be implemented by C++
components.
 *
 *  Change history:
 *                  4.20.000 - first version
 */

// Forward declare all types defined
interface ITPCC;
import "oaidl.idl";
import "ocidl.idl";

          [
                    object,
                    oleautomation,
                    uuid(FEEE6AA2-84B1-11d2-BA47-
00C04FBFE08B),
                    helpstring("ITPCC Interface"),
                    pointer_default(unique)
          ]
          interface ITPCC : IUnknown
          {

        HRESULT _stdcall NewOrder

          (

          [in] VARIANT  txn_in,

          [out] VARIANT *txn_out

          );

        HRESULT _stdcall Payment

          (

          [in] VARIANT txn_in,

          [out] VARIANT *txn_out

          );

        HRESULT _stdcall Delivery

          (

          [in] VARIANT txn_in,

          [out] VARIANT *txn_out

          );

        HRESULT _stdcall StockLevel
```

```
        (

        [in] VARIANT txn_in,

        [out] VARIANT *txn_out

        );

      HRESULT _stdcall OrderStatus

        (

        [in] VARIANT txn_in,

        [out] VARIANT *txn_out

        );

      HRESULT _stdcall CallSetComplete

        (

        );

  };    // interface ITPCC
```

tpcc_com_ps_i.c

```
/* this ALWAYS GENERATED file contains the IIDs and
CLSIDs */

/* link this file in with the server and any clients
*/


 /* File created by MIDL compiler version 6.00.0361
*/
/* at Tue Nov 10 10:51:13 2009
 */
/* Compiler settings for .\src\tpcc_com_ps.idl:
    Oicf, W1, Zp8, env=Win32 (32b run)
    protocol : dce , ms_ext, c_ext
    error checks: allocation ref bounds_check enum
stub_data
    VC __declspec() decoration level:
         __declspec(uuid()), __declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(), MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if !defined(_M_IA64) && !defined(_M_AMD64)

#pragma warning( disable: 4049 )  /* more than 64k
source lines */


#ifdef __cplusplus
extern "C"{
#endif


#include <rpc.h>
#include <rpcndr.h>

#ifdef _MIDL_USE_GUIDDEF_

#ifndef INITGUID
#define INITGUID
#include <guiddef.h>
#undef INITGUID
#else
#include <guiddef.h>
#endif

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,
b7,b8) \

DEFINE_GUID(name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8)

#else // !_MIDL_USE_GUIDDEF_

#ifndef __IID_DEFINED__
#define __IID_DEFINED__

typedef struct _IID
{
    unsigned long x;
    unsigned short s1;
    unsigned short s2;
    unsigned char  c[8];
} IID;

#endif // __IID_DEFINED__

#ifndef CLSID_DEFINED
#define CLSID_DEFINED
typedef IID CLSID;
#endif // CLSID_DEFINED

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,
b7,b8) \
        const type name =
{l,w1,w2,{b1,b2,b3,b4,b5,b6,b7,b8}}

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
IID_ITPCC,0xFEEE6AA2,0x84B1,0x11d2,0xBA,0x47,0x00,0xC
0,0x4F,0xBF,0xE0,0x8B);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif



#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/



/* this ALWAYS GENERATED file contains the IIDs and
CLSIDs */

/* link this file in with the server and any clients
*/


 /* File created by MIDL compiler version 6.00.0361
*/
/* at Tue Nov 10 10:51:13 2009
 */
/* Compiler settings for .\src\tpcc_com_ps.idl:
    Oicf, W1, Zp8, env=Win64 (32b run,appending)
    protocol : dce , ms_ext, c_ext, robust
    error checks: allocation ref bounds_check enum
stub_data
    VC __declspec() decoration level:
         __declspec(uuid()), __declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(), MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if defined(_M_IA64) || defined(_M_AMD64)


#pragma warning( disable: 4049 )  /* more than 64k
source lines */


#ifdef __cplusplus
extern "C"{
#endif


#include <rpc.h>
#include <rpcndr.h>

#ifdef _MIDL_USE_GUIDDEF_

#ifndef INITGUID
#define INITGUID
#include <guiddef.h>
#undef INITGUID
#else
#include <guiddef.h>
#endif

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,
b7,b8) \

DEFINE_GUID(name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8)
```

```
#else // !_MIDL_USE_GUIDDEF_

#ifndef __IID_DEFINED__
#define __IID_DEFINED__

typedef struct _IID
{
    unsigned long x;
    unsigned short s1;
    unsigned short s2;
    unsigned char  c[8];
} IID;

#endif // __IID_DEFINED__

#ifndef CLSID_DEFINED
#define CLSID_DEFINED
typedef IID CLSID;
#endif // CLSID_DEFINED

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,
b7,b8) \
        const type name =
{l,w1,w2,{b1,b2,b3,b4,b5,b6,b7,b8}}

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
IID_ITPCC,0xFEEE6AA2,0x84B1,0x11d2,0xBA,0x47,0x00,0xC
0,0x4F,0xBF,0xE0,0x8B);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif



#endif /* defined(_M_IA64) || defined(_M_AMD64)*/
```

tpcc_com_ps_p.c

```
/* this ALWAYS GENERATED file contains the proxy stub
code */


 /* File created by MIDL compiler version 6.00.0361
*/
/* at Tue Nov 10 10:51:13 2009
 */
/* Compiler settings for .\src\tpcc_com_ps.idl:
    Oicf, W1, Zp8, env=Win32 (32b run)
    protocol : dce , ms_ext, c_ext
    error checks: allocation ref bounds_check enum
stub_data
    VC __declspec() decoration level:
         __declspec(uuid()), __declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(), MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if !defined(_M_IA64) && !defined(_M_AMD64)


#pragma warning( disable: 4049 )  /* more than 64k
source lines */
#if _MSC_VER >= 1200
#pragma warning(push)
#endif
#pragma warning( disable: 4100 ) /* unreferenced
arguments in x86 call */
#pragma warning( disable: 4211 )  /* redefine extent
to static */
#pragma warning( disable: 4232 )  /* dllimport
identity*/
#define USE_STUBLESS_PROXY


/* verify that the <rpcproxy.h> version is high
enough to compile this file*/
#ifndef __REDQ_RPCPROXY_H_VERSION__
#define __REQUIRED_RPCPROXY_H_VERSION__ 440
#endif


#include "rpcproxy.h"
#ifndef __RPCPROXY_H_VERSION__
#error this stub requires an updated version of
<rpcproxy.h>
#endif // __RPCPROXY_H_VERSION__


#include "tpcc_com_ps.h"

#define TYPE_FORMAT_STRING_SIZE   1023
#define PROC_FORMAT_STRING_SIZE   193
#define TRANSMIT_AS_TABLE_SIZE    0
#define WIRE_MARSHAL_TABLE_SIZE   1

typedef struct _MIDL_TYPE_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[ TYPE_FORMAT_STRING_SIZE ];
    } MIDL_TYPE_FORMAT_STRING;

typedef struct _MIDL_PROC_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[ PROC_FORMAT_STRING_SIZE ];
    } MIDL_PROC_FORMAT_STRING;


static RPC_SYNTAX_IDENTIFIER  _RpcTransferSyntax =
{{0x8A885D04,0x1CEB,0x11C9,{0x9F,0xE8,0x08,0x00,0x2B,
0x10,0x48,0x60}},{2,0}};
extern const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString;
extern const MIDL_PROC_FORMAT_STRING
__MIDL_ProcFormatString;


extern const MIDL_STUB_DESC Object_StubDesc;


extern const MIDL_SERVER_INFO ITPCC_ServerInfo;
extern const MIDL_STUBLESS_PROXY_INFO
ITPCC_ProxyInfo;


extern const USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[ WIRE_MARSHAL_TABLE_SIZE ];

#if !defined(__RPC_WIN32__)
#error  Invalid build platform for this stub.
#endif

#if !(TARGET_IS_NT40_OR_LATER)
#error You need a Windows NT 4.0 or later to run this
stub because it uses these features:
#error   -Oif or -Oicf, [wire_marshal] or
[user_marshal] attribute.
#error However, your C/C++ compilation flags indicate
you intend to run this app on earlier systems.
#error This app will die there with the
RPC_X_WRONG_STUB_VERSION error.
#endif


static const MIDL_PROC_FORMAT_STRING
__MIDL_ProcFormatString =
    {
        0,
        {

          /* Procedure NewOrder */

                                0x33,               /*
FC_AUTO_HANDLE */
                                0x6c,               /*
Old Flags:  object, Oi2 */
/*  2 */  NdrFcLong( 0x0 ),   /* 0 */
/*  6 */  NdrFcShort( 0x3 ),  /* 3 */
/*  8 */  NdrFcShort( 0x1c ), /* x86 Stack
size/offset = 28 */
/* 10 */  NdrFcShort( 0x0 ),  /* 0 */
/* 12 */  NdrFcShort( 0x8 ),  /* 8 */
/* 14 */  0x7,                /* Oi2 Flags:  srv must
size, clt must size, has return, */
                                0x3,                /*
3 */

          /* Parameter txn_in */

/* 16 */  NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 18 */  NdrFcShort( 0x4 ),  /* x86 Stack
size/offset = 4 */
```

```
/* 20 */  NdrFcShort( 0x3e2 ),          /* Type
Offset=994 */

         /* Parameter txn_out */

/* 22 */  NdrFcShort( 0x4113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=16 */
/* 24 */  NdrFcShort( 0x14 ), /* x86 Stack
size/offset = 20 */
/* 26 */  NdrFcShort( 0x3f4 ),          /* Type
Offset=1012 */

         /* Return value */

/* 28 */  NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 30 */  NdrFcShort( 0x18 ), /* x86 Stack
size/offset = 24 */
/* 32 */  0x8,                /* FC_LONG */
                              0x0,                /*
0 */

         /* Procedure Payment */

/* 34 */  0x33,               /* FC_AUTO_HANDLE */
                              0x6c,               /*
Old Flags:  object, Oi2 */
/* 36 */  NdrFcLong( 0x0 ),   /* 0 */
/* 40 */  NdrFcShort( 0x4 ),  /* 4 */
/* 42 */  NdrFcShort( 0x1c ), /* x86 Stack
size/offset = 28 */
/* 44 */  NdrFcShort( 0x0 ),  /* 0 */
/* 46 */  NdrFcShort( 0x8 ),  /* 8 */
/* 48 */  0x7,                /* Oi2 Flags:  srv must
size, clt must size, has return, */
                              0x3,                /*
3 */

         /* Parameter txn_in */

/* 50 */  NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 52 */  NdrFcShort( 0x4 ),  /* x86 Stack
size/offset = 4 */
/* 54 */  NdrFcShort( 0x3e2 ),          /* Type
Offset=994 */

         /* Parameter txn_out */

/* 56 */  NdrFcShort( 0x4113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=16 */
/* 58 */  NdrFcShort( 0x14 ), /* x86 Stack
size/offset = 20 */
/* 60 */  NdrFcShort( 0x3f4 ),          /* Type
Offset=1012 */

         /* Return value */

/* 62 */  NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 64 */  NdrFcShort( 0x18 ), /* x86 Stack
size/offset = 24 */
/* 66 */  0x8,                /* FC_LONG */
                              0x0,                /*
0 */

         /* Procedure Delivery */

/* 68 */  0x33,               /* FC_AUTO_HANDLE */
                              0x6c,               /*
Old Flags:  object, Oi2 */
/* 70 */  NdrFcLong( 0x0 ),   /* 0 */
/* 74 */  NdrFcShort( 0x5 ),  /* 5 */
/* 76 */  NdrFcShort( 0x1c ), /* x86 Stack
size/offset = 28 */
/* 78 */  NdrFcShort( 0x0 ),  /* 0 */
/* 80 */  NdrFcShort( 0x8 ),  /* 8 */
/* 82 */  0x7,                /* Oi2 Flags:  srv must
size, clt must size, has return, */
                              0x3,                /*
3 */

         /* Parameter txn_in */

/* 84 */  NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 86 */  NdrFcShort( 0x4 ),  /* x86 Stack
size/offset = 4 */
/* 88 */  NdrFcShort( 0x3e2 ),          /* Type
Offset=994 */

         /* Parameter txn_out */

/* 90 */  NdrFcShort( 0x4113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=16 */
/* 92 */  NdrFcShort( 0x14 ), /* x86 Stack
size/offset = 20 */
/* 94 */  NdrFcShort( 0x3f4 ),          /* Type
Offset=1012 */

         /* Return value */

/* 96 */  NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 98 */  NdrFcShort( 0x18 ), /* x86 Stack
size/offset = 24 */
/* 100 */ 0x8,                /* FC_LONG */
                              0x0,                /*
0 */

         /* Procedure StockLevel */

/* 102 */ 0x33,               /* FC_AUTO_HANDLE */
                              0x6c,               /*
Old Flags:  object, Oi2 */
/* 104 */ NdrFcLong( 0x0 ),   /* 0 */
/* 108 */ NdrFcShort( 0x6 ),  /* 6 */
/* 110 */ NdrFcShort( 0x1c ), /* x86 Stack
size/offset = 28 */
/* 112 */ NdrFcShort( 0x0 ),  /* 0 */
/* 114 */ NdrFcShort( 0x8 ),  /* 8 */
/* 116 */ 0x7,                /* Oi2 Flags:  srv must
size, clt must size, has return, */
                              0x3,                /*
3 */

         /* Parameter txn_in */

/* 118 */ NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 120 */ NdrFcShort( 0x4 ),  /* x86 Stack
size/offset = 4 */
/* 122 */ NdrFcShort( 0x3e2 ),          /* Type
Offset=994 */

         /* Parameter txn_out */

/* 124 */ NdrFcShort( 0x4113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=16 */
/* 126 */ NdrFcShort( 0x14 ), /* x86 Stack
size/offset = 20 */
/* 128 */ NdrFcShort( 0x3f4 ),          /* Type
Offset=1012 */

         /* Return value */

/* 130 */ NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 132 */ NdrFcShort( 0x18 ), /* x86 Stack
size/offset = 24 */
/* 134 */ 0x8,                /* FC_LONG */
                              0x0,                /*
0 */

         /* Procedure OrderStatus */

/* 136 */ 0x33,               /* FC_AUTO_HANDLE */
                              0x6c,               /*
Old Flags:  object, Oi2 */
/* 138 */ NdrFcLong( 0x0 ),   /* 0 */
/* 142 */ NdrFcShort( 0x7 ),  /* 7 */
/* 144 */ NdrFcShort( 0x1c ), /* x86 Stack
size/offset = 28 */
/* 146 */ NdrFcShort( 0x0 ),  /* 0 */
/* 148 */ NdrFcShort( 0x8 ),  /* 8 */
/* 150 */ 0x7,                /* Oi2 Flags:  srv must
size, clt must size, has return, */
                              0x3,                /*
3 */

         /* Parameter txn_in */

/* 152 */ NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 154 */ NdrFcShort( 0x4 ),  /* x86 Stack
size/offset = 4 */
/* 156 */ NdrFcShort( 0x3e2 ),          /* Type
Offset=994 */

         /* Parameter txn_out */
```

```
/* 158 */ NdrFcShort( 0x4113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=16 */
/* 160 */ NdrFcShort( 0x14 ), /* x86 Stack
size/offset = 20 */
/* 162 */ NdrFcShort( 0x3f4 ),          /* Type
Offset=1012 */

         /* Return value */

/* 164 */ NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 166 */ NdrFcShort( 0x18 ), /* x86 Stack
size/offset = 24 */
/* 168 */ 0x8,                /* FC_LONG */
                              0x0,                /*
0 */

         /* Procedure CallSetComplete */

/* 170 */ 0x33,               /* FC_AUTO_HANDLE */
                              0x6c,               /*
Old Flags:  object, Oi2 */
/* 172 */ NdrFcLong( 0x0 ),   /* 0 */
/* 176 */ NdrFcShort( 0x8 ),  /* 8 */
/* 178 */ NdrFcShort( 0x8 ),  /* x86 Stack
size/offset = 8 */
/* 180 */ NdrFcShort( 0x0 ),  /* 0 */
/* 182 */ NdrFcShort( 0x8 ),  /* 8 */
/* 184 */ 0x4,                /* Oi2 Flags:  has
return, */
                              0x1,                /*
1 */

         /* Return value */

/* 186 */ NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 188 */ NdrFcShort( 0x4 ),  /* x86 Stack
size/offset = 4 */
/* 190 */ 0x8,                /* FC_LONG */
                              0x0,                /*
0 */

                              0x0
       }
    };

static const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString =
    {
        0,
        {
                              NdrFcShort( 0x0 ),  /*
0 */
/*  2 */
                              0x12, 0x0,          /*
FC_UP */
/*  4 */  NdrFcShort( 0x3ca ),          /* Offset=
970 (974) */
/*  6 */
                              0x2b,               /*
FC_NON_ENCAPSULATED_UNION */
                              0x9,                /*
FC_ULONG */
/*  8 */  0x7,                /* Corr desc: FC_USHORT
*/
                              0x0,                /*
*/
/* 10 */  NdrFcShort( 0xfff8 ),         /* -8 */
/* 12 */  NdrFcShort( 0x2 ),  /* Offset= 2 (14) */
/* 14 */  NdrFcShort( 0x10 ), /* 16 */
/* 16 */  NdrFcShort( 0x2f ), /* 47 */
/* 18 */  NdrFcLong( 0x14 ),  /* 20 */
/* 22 */  NdrFcShort( 0x800b ),         /* Simple arm
type: FC_HYPER */
/* 24 */  NdrFcLong( 0x3 ),   /* 3 */
/* 28 */  NdrFcShort( 0x8008 ),         /* Simple arm
type: FC_LONG */
/* 30 */  NdrFcLong( 0x11 ),  /* 17 */
/* 34 */  NdrFcShort( 0x8001 ),         /* Simple arm
type: FC_BYTE */
/* 36 */  NdrFcLong( 0x2 ),   /* 2 */
/* 40 */  NdrFcShort( 0x8006 ),         /* Simple arm
type: FC_SHORT */
/* 42 */  NdrFcLong( 0x4 ),   /* 4 */
/* 46 */  NdrFcShort( 0x800a ),         /* Simple arm
type: FC_FLOAT */
/* 48 */  NdrFcLong( 0x5 ),   /* 5 */
/* 52 */  NdrFcShort( 0x800c ),         /* Simple arm
type: FC_DOUBLE */
/* 54 */  NdrFcLong( 0xb ),   /* 11 */
/* 58 */  NdrFcShort( 0x8006 ),         /* Simple arm
type: FC_SHORT */
/* 60 */  NdrFcLong( 0xa ),   /* 10 */
/* 64 */  NdrFcShort( 0x8008 ),         /* Simple arm
type: FC_LONG */
/* 66 */  NdrFcLong( 0x6 ),   /* 6 */
/* 70 */  NdrFcShort( 0xe8 ), /* Offset= 232 (302) */
/* 72 */  NdrFcLong( 0x7 ),   /* 7 */
/* 76 */  NdrFcShort( 0x800c ),         /* Simple arm
type: FC_DOUBLE */
/* 78 */  NdrFcLong( 0x8 ),   /* 8 */
/* 82 */  NdrFcShort( 0xe2 ), /* Offset= 226 (308) */
/* 84 */  NdrFcLong( 0xd ),   /* 13 */
/* 88 */  NdrFcShort( 0xf4 ), /* Offset= 244 (332) */
/* 90 */  NdrFcLong( 0x9 ),   /* 9 */
/* 94 */  NdrFcShort( 0x100 ),          /* Offset=
256 (350) */
/* 96 */  NdrFcLong( 0x2000 ),          /* 8192 */
/* 100 */ NdrFcShort( 0x10c ),          /* Offset=
268 (368) */
/* 102 */ NdrFcLong( 0x24 ),  /* 36 */
/* 106 */ NdrFcShort( 0x31a ),          /* Offset=
794 (900) */
/* 108 */ NdrFcLong( 0x4024 ),          /* 16420 */
/* 112 */ NdrFcShort( 0x314 ),          /* Offset=
788 (900) */
/* 114 */ NdrFcLong( 0x4011 ),          /* 16401 */
/* 118 */ NdrFcShort( 0x312 ),          /* Offset=
786 (904) */
/* 120 */ NdrFcLong( 0x4002 ),          /* 16386 */
/* 124 */ NdrFcShort( 0x310 ),          /* Offset=
784 (908) */
/* 126 */ NdrFcLong( 0x4003 ),          /* 16387 */
/* 130 */ NdrFcShort( 0x30e ),          /* Offset=
782 (912) */
/* 132 */ NdrFcLong( 0x4014 ),          /* 16404 */
/* 136 */ NdrFcShort( 0x30c ),          /* Offset=
780 (916) */
/* 138 */ NdrFcLong( 0x4004 ),          /* 16388 */
/* 142 */ NdrFcShort( 0x30a ),          /* Offset=
778 (920) */
/* 144 */ NdrFcLong( 0x4005 ),          /* 16389 */
/* 148 */ NdrFcShort( 0x308 ),          /* Offset=
776 (924) */
/* 150 */ NdrFcLong( 0x400b ),          /* 16395 */
/* 154 */ NdrFcShort( 0x2f2 ),          /* Offset=
754 (908) */
/* 156 */ NdrFcLong( 0x400a ),          /* 16394 */
/* 160 */ NdrFcShort( 0x2f0 ),          /* Offset=
752 (912) */
/* 162 */ NdrFcLong( 0x4006 ),          /* 16390 */
/* 166 */ NdrFcShort( 0x2fa ),          /* Offset=
762 (928) */
/* 168 */ NdrFcLong( 0x4007 ),          /* 16391 */
/* 172 */ NdrFcShort( 0x2f0 ),          /* Offset=
752 (924) */
/* 174 */ NdrFcLong( 0x4008 ),          /* 16392 */
/* 178 */ NdrFcShort( 0x2f2 ),          /* Offset=
754 (932) */
/* 180 */ NdrFcLong( 0x400d ),          /* 16397 */
/* 184 */ NdrFcShort( 0x2f0 ),          /* Offset=
752 (936) */
/* 186 */ NdrFcLong( 0x4009 ),          /* 16393 */
/* 190 */ NdrFcShort( 0x2ee ),          /* Offset=
750 (940) */
/* 192 */ NdrFcLong( 0x6000 ),          /* 24576 */
/* 196 */ NdrFcShort( 0x2ec ),          /* Offset=
748 (944) */
/* 198 */ NdrFcLong( 0x400c ),          /* 16396 */
/* 202 */ NdrFcShort( 0x2ea ),          /* Offset=
746 (948) */
/* 204 */ NdrFcLong( 0x10 ),  /* 16 */
/* 208 */ NdrFcShort( 0x8002 ),         /* Simple arm
type: FC_CHAR */
/* 210 */ NdrFcLong( 0x12 ),  /* 18 */
/* 214 */ NdrFcShort( 0x8006 ),         /* Simple arm
type: FC_SHORT */
/* 216 */ NdrFcLong( 0x13 ),  /* 19 */
/* 220 */ NdrFcShort( 0x8008 ),         /* Simple arm
type: FC_LONG */
/* 222 */ NdrFcLong( 0x15 ),  /* 21 */
/* 226 */ NdrFcShort( 0x800b ),         /* Simple arm
type: FC_HYPER */
/* 228 */ NdrFcLong( 0x16 ),  /* 22 */
/* 232 */ NdrFcShort( 0x8008 ),         /* Simple arm
type: FC_LONG */
/* 234 */ NdrFcLong( 0x17 ),  /* 23 */
/* 238 */ NdrFcShort( 0x8008 ),         /* Simple arm
type: FC_LONG */
/* 240 */ NdrFcLong( 0xe ),   /* 14 */
/* 244 */ NdrFcShort( 0x2c8 ),          /* Offset=
712 (956) */
/* 246 */ NdrFcLong( 0x400e ),          /* 16398 */
/* 250 */ NdrFcShort( 0x2cc ),          /* Offset=
716 (966) */
/* 252 */ NdrFcLong( 0x4010 ),          /* 16400 */
```

```
/* 256 */ NdrFcShort( 0x2ca ),          /* Offset=
714 (970) */
/* 258 */ NdrFcLong( 0x4012 ),          /* 16402 */
/* 262 */ NdrFcShort( 0x286 ),          /* Offset=
646 (908) */
/* 264 */ NdrFcLong( 0x4013 ),          /* 16403 */
/* 268 */ NdrFcShort( 0x284 ),          /* Offset=
644 (912) */
/* 270 */ NdrFcLong( 0x4015 ),          /* 16405 */
/* 274 */ NdrFcShort( 0x282 ),          /* Offset=
642 (916) */
/* 276 */ NdrFcLong( 0x4016 ),          /* 16406 */
/* 280 */ NdrFcShort( 0x278 ),          /* Offset=
632 (912) */
/* 282 */ NdrFcLong( 0x4017 ),          /* 16407 */
/* 286 */ NdrFcShort( 0x272 ),          /* Offset=
626 (912) */
/* 288 */ NdrFcLong( 0x0 ),   /* 0 */
/* 292 */ NdrFcShort( 0x0 ),  /* Offset= 0 (292) */
/* 294 */ NdrFcLong( 0x1 ),   /* 1 */
/* 298 */ NdrFcShort( 0x0 ),  /* Offset= 0 (298) */
/* 300 */ NdrFcShort( 0xffff ),         /* Offset= -1
(299) */
/* 302 */
                           0x15,               /*
FC_STRUCT */
                           0x7,                /*
7 */
/* 304 */ NdrFcShort( 0x8 ),  /* 8 */
/* 306 */ 0xb,                /* FC_HYPER */
                           0x5b,               /*
FC_END */
/* 308 */
                           0x12, 0x0,          /*
FC_UP */
/* 310 */ NdrFcShort( 0xc ),  /* Offset= 12 (322) */
/* 312 */
                           0x1b,               /*
FC_CARRAY */
                           0x1,                /*
1 */
/* 314 */ NdrFcShort( 0x2 ),  /* 2 */
/* 316 */ 0x9,                /* Corr desc: FC_ULONG
*/
                           0x0,                /*
*/
/* 318 */ NdrFcShort( 0xfffc ),         /* -4 */
/* 320 */ 0x6,                /* FC_SHORT */
                           0x5b,               /*
FC_END */
/* 322 */
                           0x17,               /*
FC_CSTRUCT */
                           0x3,                /*
3 */
/* 324 */ NdrFcShort( 0x8 ),  /* 8 */
/* 326 */ NdrFcShort( 0xfff2 ),         /* Offset= -
14 (312) */
/* 328 */ 0x8,                /* FC_LONG */
                           0x8,                /*
FC_LONG */
/* 330 */ 0x5c,               /* FC_PAD */
                           0x5b,               /*
FC_END */
/* 332 */
                           0x2f,               /*
FC_IP */
                           0x5a,               /*
FC_CONSTANT_IID */
/* 334 */ NdrFcLong( 0x0 ),   /* 0 */
/* 338 */ NdrFcShort( 0x0 ),  /* 0 */
/* 340 */ NdrFcShort( 0x0 ),  /* 0 */
/* 342 */ 0xc0,               /* 192 */
                           0x0,                /*
0 */
/* 344 */ 0x0,                /* 0 */
                           0x0,                /*
0 */
/* 346 */ 0x0,                /* 0 */
                           0x0,                /*
0 */
/* 348 */ 0x0,                /* 0 */
                           0x46,               /*
70 */
/* 350 */
                           0x2f,               /*
FC_IP */
                           0x5a,               /*
FC_CONSTANT_IID */
/* 352 */ NdrFcLong( 0x20400 ),         /* 132096 */
/* 356 */ NdrFcShort( 0x0 ),  /* 0 */
/* 358 */ NdrFcShort( 0x0 ),  /* 0 */
/* 360 */ 0xc0,               /* 192 */
                           0x0,                /*
0 */
/* 362 */ 0x0,                /* 0 */
                           0x0,                /*
0 */
/* 364 */ 0x0,                /* 0 */
                           0x0,                /*
0 */
/* 366 */ 0x0,                /* 0 */
                           0x46,               /*
70 */
/* 368 */
                           0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 370 */ NdrFcShort( 0x2 ),  /* Offset= 2 (372) */
/* 372 */
                           0x12, 0x0,          /*
FC_UP */
/* 374 */ NdrFcShort( 0x1fc ),          /* Offset=
508 (882) */
/* 376 */
                           0x2a,               /*
FC_ENCAPSULATED_UNION */
                           0x49,               /*
73 */
/* 378 */ NdrFcShort( 0x18 ), /* 24 */
/* 380 */ NdrFcShort( 0xa ),  /* 10 */
/* 382 */ NdrFcLong( 0x8 ),   /* 8 */
/* 386 */ NdrFcShort( 0x58 ), /* Offset= 88 (474) */
/* 388 */ NdrFcLong( 0xd ),   /* 13 */
/* 392 */ NdrFcShort( 0x78 ), /* Offset= 120 (512) */
/* 394 */ NdrFcLong( 0x9 ),   /* 9 */
/* 398 */ NdrFcShort( 0x94 ), /* Offset= 148 (546) */
/* 400 */ NdrFcLong( 0xc ),   /* 12 */
/* 404 */ NdrFcShort( 0xbc ), /* Offset= 188 (592) */
/* 406 */ NdrFcLong( 0x24 ),  /* 36 */
/* 410 */ NdrFcShort( 0x114 ),          /* Offset=
276 (686) */
/* 412 */ NdrFcLong( 0x800d ),          /* 32781 */
/* 416 */ NdrFcShort( 0x130 ),          /* Offset=
304 (720) */
/* 418 */ NdrFcLong( 0x10 ),  /* 16 */
/* 422 */ NdrFcShort( 0x148 ),          /* Offset=
328 (750) */
/* 424 */ NdrFcLong( 0x2 ),   /* 2 */
/* 428 */ NdrFcShort( 0x160 ),          /* Offset=
352 (780) */
/* 430 */ NdrFcLong( 0x3 ),   /* 3 */
/* 434 */ NdrFcShort( 0x178 ),          /* Offset=
376 (810) */
/* 436 */ NdrFcLong( 0x14 ),  /* 20 */
/* 440 */ NdrFcShort( 0x190 ),          /* Offset=
400 (840) */
/* 442 */ NdrFcShort( 0xffff ),         /* Offset= -1
(441) */
/* 444 */
                           0x1b,               /*
FC_CARRAY */
                           0x3,                /*
3 */
/* 446 */ NdrFcShort( 0x4 ),  /* 4 */
/* 448 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                           0x0,                /*
*/
/* 450 */ NdrFcShort( 0x0 ),  /* 0 */
/* 452 */
                           0x4b,               /*
FC_PP */
                           0x5c,               /*
FC_PAD */
/* 454 */
                           0x48,               /*
FC_VARIABLE_REPEAT */
                           0x49,               /*
FC_FIXED_OFFSET */
/* 456 */ NdrFcShort( 0x4 ),  /* 4 */
/* 458 */ NdrFcShort( 0x0 ),  /* 0 */
/* 460 */ NdrFcShort( 0x1 ),  /* 1 */
/* 462 */ NdrFcShort( 0x0 ),  /* 0 */
/* 464 */ NdrFcShort( 0x0 ),  /* 0 */
/* 466 */ 0x12, 0x0,          /* FC_UP */
/* 468 */ NdrFcShort( 0xff6e ),         /* Offset= -
146 (322) */
/* 470 */
                           0x5b,               /*
FC_END */
                           0x8,                /*
FC_LONG */
/* 472 */ 0x5c,               /* FC_PAD */
                           0x5b,               /*
FC_END */
/* 474 */
```

```
                            0x16,               /*
FC_PSTRUCT */
                            0x3,                /*
3 */
/* 476 */ NdrFcShort( 0x8 ),  /* 8 */
/* 478 */
                            0x4b,               /*
FC_PP */
                            0x5c,               /*
FC_PAD */
/* 480 */
                            0x46,               /*
FC_NO_REPEAT */
                            0x5c,               /*
FC_PAD */
/* 482 */ NdrFcShort( 0x4 ),  /* 4 */
/* 484 */ NdrFcShort( 0x4 ),  /* 4 */
/* 486 */ 0x11, 0x0,          /* FC_RP */
/* 488 */ NdrFcShort( 0xffd4 ),         /* Offset= -
44 (444) */
/* 490 */
                            0x5b,               /*
FC_END */

                            0x8,                /*
FC_LONG */
/* 492 */ 0x8,                /* FC_LONG */
                            0x5b,               /*
FC_END */
/* 494 */
                            0x21,               /*
FC_BOGUS_ARRAY */
                            0x3,                /*
3 */
/* 496 */ NdrFcShort( 0x0 ),  /* 0 */
/* 498 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 500 */ NdrFcShort( 0x0 ),  /* 0 */
/* 502 */ NdrFcLong( 0xffffffff ),      /* -1 */
/* 506 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 508 */ NdrFcShort( 0xff50 ),         /* Offset= -
176 (332) */
/* 510 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 512 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 514 */ NdrFcShort( 0x8 ),  /* 8 */
/* 516 */ NdrFcShort( 0x0 ),  /* 0 */
/* 518 */ NdrFcShort( 0x6 ),  /* Offset= 6 (524) */
/* 520 */ 0x8,                /* FC_LONG */
                            0x36,               /*
FC_POINTER */
/* 522 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 524 */
                            0x11, 0x0,          /*
FC_RP */
/* 526 */ NdrFcShort( 0xffe0 ),         /* Offset= -
32 (494) */
/* 528 */
                            0x21,               /*
FC_BOGUS_ARRAY */
                            0x3,                /*
3 */
/* 530 */ NdrFcShort( 0x0 ),  /* 0 */
/* 532 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 534 */ NdrFcShort( 0x0 ),  /* 0 */
/* 536 */ NdrFcLong( 0xffffffff ),      /* -1 */
/* 540 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 542 */ NdrFcShort( 0xff40 ),         /* Offset= -
192 (350) */
/* 544 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 546 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 548 */ NdrFcShort( 0x8 ),  /* 8 */
/* 550 */ NdrFcShort( 0x0 ),  /* 0 */
/* 552 */ NdrFcShort( 0x6 ),  /* Offset= 6 (558) */
/* 554 */ 0x8,                /* FC_LONG */
                            0x36,               /*
FC_POINTER */
/* 556 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 558 */
                            0x11, 0x0,          /*
FC_RP */
/* 560 */ NdrFcShort( 0xffe0 ),         /* Offset= -
32 (528) */
/* 562 */
                            0x1b,               /*
FC_CARRAY */
                            0x3,                /*
3 */
/* 564 */ NdrFcShort( 0x4 ),  /* 4 */
/* 566 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 568 */ NdrFcShort( 0x0 ),  /* 0 */
/* 570 */
                            0x4b,               /*
FC_PP */
                            0x5c,               /*
FC_PAD */
/* 572 */
                            0x48,               /*
FC_VARIABLE_REPEAT */
                            0x49,               /*
FC_FIXED_OFFSET */
/* 574 */ NdrFcShort( 0x4 ),  /* 4 */
/* 576 */ NdrFcShort( 0x0 ),  /* 0 */
/* 578 */ NdrFcShort( 0x1 ),  /* 1 */
/* 580 */ NdrFcShort( 0x0 ),  /* 0 */
/* 582 */ NdrFcShort( 0x0 ),  /* 0 */
/* 584 */ 0x12, 0x0,          /* FC_UP */
/* 586 */ NdrFcShort( 0x184 ),          /* Offset=
388 (974) */
/* 588 */
                            0x5b,               /*
FC_END */

                            0x8,                /*
FC_LONG */
/* 590 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 592 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 594 */ NdrFcShort( 0x8 ),  /* 8 */
/* 596 */ NdrFcShort( 0x0 ),  /* 0 */
/* 598 */ NdrFcShort( 0x6 ),  /* Offset= 6 (604) */
/* 600 */ 0x8,                /* FC_LONG */
                            0x36,               /*
FC_POINTER */
/* 602 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 604 */
                            0x11, 0x0,          /*
FC_RP */
/* 606 */ NdrFcShort( 0xffd4 ),         /* Offset= -
44 (562) */
/* 608 */
                            0x2f,               /*
FC_IP */
                            0x5a,               /*
FC_CONSTANT_IID */
/* 610 */ NdrFcLong( 0x2f ),  /* 47 */
/* 614 */ NdrFcShort( 0x0 ),  /* 0 */
/* 616 */ NdrFcShort( 0x0 ),  /* 0 */
/* 618 */ 0xc0,               /* 192 */
                            0x0,                /*
0 */
/* 620 */ 0x0,                /* 0 */
                            0x0,                /*
0 */
/* 622 */ 0x0,                /* 0 */
                            0x0,                /*
0 */
/* 624 */ 0x0,                /* 0 */
                            0x46,               /*
70 */
/* 626 */
```

```
                            0x1b,               /*
FC_CARRAY */
                            0x0,                /*
0 */
/* 628 */ NdrFcShort( 0x1 ),  /* 1 */
/* 630 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 632 */ NdrFcShort( 0x4 ),  /* 4 */
/* 634 */ 0x1,                /* FC_BYTE */
                            0x5b,               /*
FC_END */
/* 636 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 638 */ NdrFcShort( 0x10 ), /* 16 */
/* 640 */ NdrFcShort( 0x0 ),  /* 0 */
/* 642 */ NdrFcShort( 0xa ),  /* Offset= 10 (652) */
/* 644 */ 0x8,                /* FC_LONG */
                            0x8,                /*
FC_LONG */
/* 646 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 648 */ NdrFcShort( 0xffd8 ),         /* Offset= -
40 (608) */
/* 650 */ 0x36,               /* FC_POINTER */
                            0x5b,               /*
FC_END */
/* 652 */
                            0x12, 0x0,          /*
FC_UP */
/* 654 */ NdrFcShort( 0xffe4 ),         /* Offset= -
28 (626) */
/* 656 */
                            0x1b,               /*
FC_CARRAY */
                            0x3,                /*
3 */
/* 658 */ NdrFcShort( 0x4 ),  /* 4 */
/* 660 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 662 */ NdrFcShort( 0x0 ),  /* 0 */
/* 664 */
                            0x4b,               /*
FC_PP */
                            0x5c,               /*
FC_PAD */
/* 666 */
                            0x48,               /*
FC_VARIABLE_REPEAT */
                            0x49,               /*
FC_FIXED_OFFSET */
/* 668 */ NdrFcShort( 0x4 ),  /* 4 */
/* 670 */ NdrFcShort( 0x0 ),  /* 0 */
/* 672 */ NdrFcShort( 0x1 ),  /* 1 */
/* 674 */ NdrFcShort( 0x0 ),  /* 0 */
/* 676 */ NdrFcShort( 0x0 ),  /* 0 */
/* 678 */ 0x12, 0x0,          /* FC_UP */
/* 680 */ NdrFcShort( 0xffd4 ),         /* Offset= -
44 (636) */
/* 682 */
                            0x5b,               /*
FC_END */
                            0x8,                /*
FC_LONG */
/* 684 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 686 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 688 */ NdrFcShort( 0x8 ),  /* 8 */
/* 690 */ NdrFcShort( 0x0 ),  /* 0 */
/* 692 */ NdrFcShort( 0x6 ),  /* Offset= 6 (698) */
/* 694 */ 0x8,                /* FC_LONG */
                            0x36,               /*
FC_POINTER */
/* 696 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 698 */
                            0x11, 0x0,          /*
FC_RP */
/* 700 */ NdrFcShort( 0xffd4 ),         /* Offset= -
44 (656) */
/* 702 */
                            0x1d,               /*
FC_SMFARRAY */
                            0x0,                /*
0 */
/* 704 */ NdrFcShort( 0x8 ),  /* 8 */
/* 706 */ 0x1,                /* FC_BYTE */
                            0x5b,               /*
FC_END */
/* 708 */
                            0x15,               /*
FC_STRUCT */
                            0x3,                /*
3 */
/* 710 */ NdrFcShort( 0x10 ), /* 16 */
/* 712 */ 0x8,                /* FC_LONG */
                            0x6,                /*
FC_SHORT */
/* 714 */ 0x6,                /* FC_SHORT */
                            0x4c,               /*
FC_EMBEDDED_COMPLEX */
/* 716 */ 0x0,                /* 0 */
                            NdrFcShort( 0xfff1 ),
          /* Offset= -15 (702) */
                            0x5b,               /*
FC_END */
/* 720 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 722 */ NdrFcShort( 0x18 ), /* 24 */
/* 724 */ NdrFcShort( 0x0 ),  /* 0 */
/* 726 */ NdrFcShort( 0xa ),  /* Offset= 10 (736) */
/* 728 */ 0x8,                /* FC_LONG */
                            0x36,               /*
FC_POINTER */
/* 730 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 732 */ NdrFcShort( 0xffe8 ),         /* Offset= -
24 (708) */
/* 734 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 736 */
                            0x11, 0x0,          /*
FC_RP */
/* 738 */ NdrFcShort( 0xff0c ),         /* Offset= -
244 (494) */
/* 740 */
                            0x1b,               /*
FC_CARRAY */
                            0x0,                /*
0 */
/* 742 */ NdrFcShort( 0x1 ),  /* 1 */
/* 744 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 746 */ NdrFcShort( 0x0 ),  /* 0 */
/* 748 */ 0x1,                /* FC_BYTE */
                            0x5b,               /*
FC_END */
/* 750 */
                            0x16,               /*
FC_PSTRUCT */
                            0x3,                /*
3 */
/* 752 */ NdrFcShort( 0x8 ),  /* 8 */
/* 754 */
                            0x4b,               /*
FC_PP */
                            0x5c,               /*
FC_PAD */
/* 756 */
                            0x46,               /*
FC_NO_REPEAT */
                            0x5c,               /*
FC_PAD */
/* 758 */ NdrFcShort( 0x4 ),  /* 4 */
/* 760 */ NdrFcShort( 0x4 ),  /* 4 */
/* 762 */ 0x12, 0x0,          /* FC_UP */
/* 764 */ NdrFcShort( 0xffe8 ),         /* Offset= -
24 (740) */
/* 766 */
                            0x5b,               /*
FC_END */
                            0x8,                /*
FC_LONG */
/* 768 */ 0x8,                /* FC_LONG */
```

```
                            0x5b,               /*
FC_END */
/* 770 */
                            0x1b,               /*
FC_CARRAY */
                            0x1,                /*
1 */
/* 772 */ NdrFcShort( 0x2 ),  /* 2 */
/* 774 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 776 */ NdrFcShort( 0x0 ),  /* 0 */
/* 778 */ 0x6,                /* FC_SHORT */
                            0x5b,               /*
FC_END */
/* 780 */
                            0x16,               /*
FC_PSTRUCT */
                            0x3,                /*
3 */
/* 782 */ NdrFcShort( 0x8 ),  /* 8 */
/* 784 */
                            0x4b,               /*
FC_PP */
                            0x5c,               /*
FC_PAD */
/* 786 */
                            0x46,               /*
FC_NO_REPEAT */
                            0x5c,               /*
FC_PAD */
/* 788 */ NdrFcShort( 0x4 ),  /* 4 */
/* 790 */ NdrFcShort( 0x4 ),  /* 4 */
/* 792 */ 0x12, 0x0,          /* FC_UP */
/* 794 */ NdrFcShort( 0xffe8 ),         /* Offset= -
24 (770) */
/* 796 */
                            0x5b,               /*
FC_END */

                            0x8,                /*
FC_LONG */
/* 798 */ 0x8,                /* FC_LONG */
                            0x5b,               /*
FC_END */
/* 800 */
                            0x1b,               /*
FC_CARRAY */
                            0x3,                /*
3 */
/* 802 */ NdrFcShort( 0x4 ),  /* 4 */
/* 804 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 806 */ NdrFcShort( 0x0 ),  /* 0 */
/* 808 */ 0x8,                /* FC_LONG */
                            0x5b,               /*
FC_END */
/* 810 */
                            0x16,               /*
FC_PSTRUCT */
                            0x3,                /*
3 */
/* 812 */ NdrFcShort( 0x8 ),  /* 8 */
/* 814 */
                            0x4b,               /*
FC_PP */
                            0x5c,               /*
FC_PAD */
/* 816 */
                            0x46,               /*
FC_NO_REPEAT */
                            0x5c,               /*
FC_PAD */
/* 818 */ NdrFcShort( 0x4 ),  /* 4 */
/* 820 */ NdrFcShort( 0x4 ),  /* 4 */
/* 822 */ 0x12, 0x0,          /* FC_UP */
/* 824 */ NdrFcShort( 0xffe8 ),         /* Offset= -
24 (800) */
/* 826 */
                            0x5b,               /*
FC_END */

                            0x8,                /*
FC_LONG */
/* 828 */ 0x8,                /* FC_LONG */
                            0x5b,               /*
FC_END */
/* 830 */
                            0x1b,               /*
FC_CARRAY */
                            0x7,                /*
7 */
/* 832 */ NdrFcShort( 0x8 ),  /* 8 */
/* 834 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 836 */ NdrFcShort( 0x0 ),  /* 0 */
/* 838 */ 0xb,                /* FC_HYPER */
                            0x5b,               /*
FC_END */
/* 840 */
                            0x16,               /*
FC_PSTRUCT */
                            0x3,                /*
3 */
/* 842 */ NdrFcShort( 0x8 ),  /* 8 */
/* 844 */
                            0x4b,               /*
FC_PP */
                            0x5c,               /*
FC_PAD */
/* 846 */
                            0x46,               /*
FC_NO_REPEAT */
                            0x5c,               /*
FC_PAD */
/* 848 */ NdrFcShort( 0x4 ),  /* 4 */
/* 850 */ NdrFcShort( 0x4 ),  /* 4 */
/* 852 */ 0x12, 0x0,          /* FC_UP */
/* 854 */ NdrFcShort( 0xffe8 ),         /* Offset= -
24 (830) */
/* 856 */
                            0x5b,               /*
FC_END */

                            0x8,                /*
FC_LONG */
/* 858 */ 0x8,                /* FC_LONG */
                            0x5b,               /*
FC_END */
/* 860 */
                            0x15,               /*
FC_STRUCT */
                            0x3,                /*
3 */
/* 862 */ NdrFcShort( 0x8 ),  /* 8 */
/* 864 */ 0x8,                /* FC_LONG */
                            0x8,                /*
FC_LONG */
/* 866 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 868 */
                            0x1b,               /*
FC_CARRAY */
                            0x3,                /*
3 */
/* 870 */ NdrFcShort( 0x8 ),  /* 8 */
/* 872 */ 0x7,                /* Corr desc: FC_USHORT
*/
                            0x0,                /*
*/
/* 874 */ NdrFcShort( 0xffd8 ),         /* -40 */
/* 876 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 878 */ NdrFcShort( 0xffee ),         /* Offset= -
18 (860) */
/* 880 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 882 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 884 */ NdrFcShort( 0x28 ), /* 40 */
/* 886 */ NdrFcShort( 0xffee ),         /* Offset= -
18 (868) */
/* 888 */ NdrFcShort( 0x0 ),  /* Offset= 0 (888) */
/* 890 */ 0x6,                /* FC_SHORT */
                            0x6,                /*
FC_SHORT */
/* 892 */ 0x8,                /* FC_LONG */
                            0x8,                /*
FC_LONG */
/* 894 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 896 */ NdrFcShort( 0xfdf8 ),         /* Offset= -
520 (376) */
/* 898 */ 0x5c,               /* FC_PAD */
```

```
                            0x5b,               /*
FC_END */
/* 900 */
                            0x12, 0x0,          /*
FC_UP */
/* 902 */ NdrFcShort( 0xfef6 ),         /* Offset= -
266 (636) */
/* 904 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 906 */ 0x1,              /* FC_BYTE */
                            0x5c,               /*
FC_PAD */
/* 908 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 910 */ 0x6,              /* FC_SHORT */
                            0x5c,               /*
FC_PAD */
/* 912 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 914 */ 0x8,              /* FC_LONG */
                            0x5c,               /*
FC_PAD */
/* 916 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 918 */ 0xb,              /* FC_HYPER */
                            0x5c,               /*
FC_PAD */
/* 920 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 922 */ 0xa,              /* FC_FLOAT */
                            0x5c,               /*
FC_PAD */
/* 924 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 926 */ 0xc,              /* FC_DOUBLE */
                            0x5c,               /*
FC_PAD */
/* 928 */
                            0x12, 0x0,          /*
FC_UP */
/* 930 */ NdrFcShort( 0xfd8c ),         /* Offset= -
628 (302) */
/* 932 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 934 */ NdrFcShort( 0xfd8e ),         /* Offset= -
626 (308) */
/* 936 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 938 */ NdrFcShort( 0xfda2 ),         /* Offset= -
606 (332) */
/* 940 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 942 */ NdrFcShort( 0xfdb0 ),         /* Offset= -
592 (350) */
/* 944 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 946 */ NdrFcShort( 0xfdbe ),         /* Offset= -
578 (368) */
/* 948 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 950 */ NdrFcShort( 0x2 ),  /* Offset= 2 (952) */
/* 952 */
                            0x12, 0x0,          /*
FC_UP */
/* 954 */ NdrFcShort( 0x14 ), /* Offset= 20 (974) */
/* 956 */
                            0x15,               /*
FC_STRUCT */
                            0x7,                /*
7 */
/* 958 */ NdrFcShort( 0x10 ), /* 16 */
/* 960 */ 0x6,              /* FC_SHORT */
                            0x1,                /*
FC_BYTE */
/* 962 */ 0x1,              /* FC_BYTE */
                            0x8,                /*
FC_LONG */
/* 964 */ 0xb,              /* FC_HYPER */
                            0x5b,               /*
FC_END */
/* 966 */
                            0x12, 0x0,          /*
FC_UP */
/* 968 */ NdrFcShort( 0xfff4 ),         /* Offset= -
12 (956) */
/* 970 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 972 */ 0x2,              /* FC_CHAR */
                            0x5c,               /*
FC_PAD */
/* 974 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x7,                /*
7 */
/* 976 */ NdrFcShort( 0x20 ), /* 32 */
/* 978 */ NdrFcShort( 0x0 ),  /* 0 */
/* 980 */ NdrFcShort( 0x0 ),  /* Offset= 0 (980) */
/* 982 */ 0x8,              /* FC_LONG */
                            0x8,                /*
FC_LONG */
/* 984 */ 0x6,              /* FC_SHORT */
                            0x6,                /*
FC_SHORT */
/* 986 */ 0x6,              /* FC_SHORT */
                            0x6,                /*
FC_SHORT */
/* 988 */ 0x4c,             /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 990 */ NdrFcShort( 0xfc28 ),         /* Offset= -
984 (6) */
/* 992 */ 0x5c,             /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 994 */ 0xb4,             /* FC_USER_MARSHAL */
                            0x83,               /*
131 */
/* 996 */ NdrFcShort( 0x0 ),  /* 0 */
/* 998 */ NdrFcShort( 0x10 ), /* 16 */
/* 1000 */          NdrFcShort( 0x0 ),  /* 0 */
/* 1002 */          NdrFcShort( 0xfc18 ),       /*
Offset= -1000 (2) */
/* 1004 */
                            0x11, 0x4,          /*
FC_RP [alloced_on_stack] */
/* 1006 */          NdrFcShort( 0x6 ),  /* Offset= 6
(1012) */
/* 1008 */
                            0x13, 0x0,          /*
FC_OP */
/* 1010 */          NdrFcShort( 0xffdc ),       /*
Offset= -36 (974) */
/* 1012 */          0xb4,               /*
FC_USER_MARSHAL */
                            0x83,               /*
131 */
/* 1014 */          NdrFcShort( 0x0 ),  /* 0 */
/* 1016 */          NdrFcShort( 0x10 ), /* 16 */
/* 1018 */          NdrFcShort( 0x0 ),  /* 0 */
/* 1020 */          NdrFcShort( 0xfff4 ),       /*
Offset= -12 (1008) */
                            0x0
        }
    };

static const USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[ WIRE_MARSHAL_TABLE_SIZE ] =
        {

            {
            VARIANT_UserSize
            ,VARIANT_UserMarshal
            ,VARIANT_UserUnmarshal
            ,VARIANT_UserFree
            }

        };



/* Standard interface: __MIDL_itf_tpcc_com_ps_0000,
ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00}} */


/* Object interface: IUnknown, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0
x00,0x00,0x00,0x46}} */


/* Object interface: ITPCC, ver. 0.0,
```

```
GUID={0xFEEE6AA2,0x84B1,0x11d2,{0xBA,0x47,0x00,0xC0,0
x4F,0xBF,0xE0,0x8B}} */

#pragma code_seg(".orpc")
static const unsigned short
ITPCC_FormatStringOffsetTable[] =
    {
    0,
    34,
    68,
    102,
    136,
    170
    };

static const MIDL_STUBLESS_PROXY_INFO ITPCC_ProxyInfo
=
    {
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &ITPCC_FormatStringOffsetTable[-3],
    0,
    0,
    0
    };


static const MIDL_SERVER_INFO ITPCC_ServerInfo =
    {
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &ITPCC_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0};
CINTERFACE_PROXY_VTABLE(9) _ITPCCProxyVtbl =
{
    &ITPCC_ProxyInfo,
    &IID_ITPCC,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    (void *) (INT_PTR) -1 /* ITPCC::NewOrder */ ,
    (void *) (INT_PTR) -1 /* ITPCC::Payment */ ,
    (void *) (INT_PTR) -1 /* ITPCC::Delivery */ ,
    (void *) (INT_PTR) -1 /* ITPCC::StockLevel */ ,
    (void *) (INT_PTR) -1 /* ITPCC::OrderStatus */ ,
    (void *) (INT_PTR) -1 /* ITPCC::CallSetComplete
*/
};

const CInterfaceStubVtbl _ITPCCStubVtbl =
{
    &IID_ITPCC,
    &ITPCC_ServerInfo,
    9,
    0, /* pure interpreted */
    CStdStubBuffer_METHODS
};
```

```
static const MIDL_STUB_DESC Object_StubDesc =
    {
    0,
    NdrOleAllocate,
    NdrOleFree,
    0,
    0,
    0,
    0,
    0,
    __MIDL_TypeFormatString.Format,
    1, /* -error bounds_check flag */
    0x20000, /* Ndr library version */
    0,
    0x6000169, /* MIDL Version 6.0.361 */
    0,
    UserMarshalRoutines,
    0,  /* notify & notify_flag routine table */
    0x1, /* MIDL flag */
    0, /* cs routines */
    0,   /* proxy/server info */
    0   /* Reserved5 */
    };

const CInterfaceProxyVtbl *
_tpcc_com_ps_ProxyVtblList[] =
{
    ( CInterfaceProxyVtbl *) &_ITPCCProxyVtbl,
    0
};

const CInterfaceStubVtbl *
_tpcc_com_ps_StubVtblList[] =
{
    ( CInterfaceStubVtbl *) &_ITPCCStubVtbl,
    0
};

PCInterfaceName const
_tpcc_com_ps_InterfaceNamesList[] =
{
    "ITPCC",
    0
};


#define _tpcc_com_ps_CHECK_IID(n)
          IID_GENERIC_CHECK_IID( _tpcc_com_ps, pIID,
n)

int __stdcall _tpcc_com_ps_IID_Lookup( const IID *
pIID, int * pIndex )
{

    if(!_tpcc_com_ps_CHECK_IID(0))
        {
        *pIndex = 0;
        return 1;
        }

    return 0;
}
```

```
const ExtendedProxyFileInfo tpcc_com_ps_ProxyFileInfo
=
{
    (PCInterfaceProxyVtblList *) &
_tpcc_com_ps_ProxyVtblList,
    (PCInterfaceStubVtblList *) &
_tpcc_com_ps_StubVtblList,
    (const PCInterfaceName * ) &
_tpcc_com_ps_InterfaceNamesList,
    0, // no delegation
    & _tpcc_com_ps_IID_Lookup,
    1,
    2,
    0, /* table of [async_uuid] interfaces */
    0, /* Filler1 */
    0, /* Filler2 */
    0  /* Filler3 */
};
#if _MSC_VER >= 1200
#pragma warning(pop)
#endif


#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/



/* this ALWAYS GENERATED file contains the proxy stub
code */


 /* File created by MIDL compiler version 6.00.0361
*/
/* at Tue Nov 10 10:51:13 2009
 */
/* Compiler settings for .\src\tpcc_com_ps.idl:
    Oicf, W1, Zp8, env=Win64 (32b run,appending)
    protocol : dce , ms_ext, c_ext, robust
    error checks: allocation ref bounds_check enum
stub_data
    VC __declspec() decoration level:
         __declspec(uuid()), __declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(), MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if defined(_M_IA64) || defined(_M_AMD64)


#pragma warning( disable: 4049 )  /* more than 64k
source lines */
#if _MSC_VER >= 1200
#pragma warning(push)
#endif

#pragma warning( disable: 4211 )  /* redefine extent
to static */
#pragma warning( disable: 4232 )  /* dllimport
identity*/
#define USE_STUBLESS_PROXY
```

```
/* verify that the <rpcproxy.h> version is high
enough to compile this file*/
#ifndef __REDQ_RPCPROXY_H_VERSION__
#define __REQUIRED_RPCPROXY_H_VERSION__ 475
#endif


#include "rpcproxy.h"
#ifndef __RPCPROXY_H_VERSION__
#error this stub requires an updated version of
<rpcproxy.h>
#endif // __RPCPROXY_H_VERSION__


#include "tpcc_com_ps.h"

#define TYPE_FORMAT_STRING_SIZE   1003
#define PROC_FORMAT_STRING_SIZE   253
#define TRANSMIT_AS_TABLE_SIZE    0
#define WIRE_MARSHAL_TABLE_SIZE   1

typedef struct _MIDL_TYPE_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[ TYPE_FORMAT_STRING_SIZE ];
    } MIDL_TYPE_FORMAT_STRING;

typedef struct _MIDL_PROC_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[ PROC_FORMAT_STRING_SIZE ];
    } MIDL_PROC_FORMAT_STRING;


static RPC_SYNTAX_IDENTIFIER  _RpcTransferSyntax =
{{0x8A885D04,0x1CEB,0x11C9,{0x9F,0xE8,0x08,0x00,0x2B,
0x10,0x48,0x60}},{2,0}};


extern const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString;
extern const MIDL_PROC_FORMAT_STRING
__MIDL_ProcFormatString;


extern const MIDL_STUB_DESC Object_StubDesc;


extern const MIDL_SERVER_INFO ITPCC_ServerInfo;
extern const MIDL_STUBLESS_PROXY_INFO
ITPCC_ProxyInfo;


extern const USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[ WIRE_MARSHAL_TABLE_SIZE ];

#if !defined(__RPC_WIN64__)
#error  Invalid build platform for this stub.
#endif

static const MIDL_PROC_FORMAT_STRING
__MIDL_ProcFormatString =
    {
        0,
        {

          /* Procedure NewOrder */

                                0x33,               /*
FC_AUTO_HANDLE */
                                0x6c,               /*
Old Flags:  object, Oi2 */
/*  2 */  NdrFcLong( 0x0 ),   /* 0 */
/*  6 */  NdrFcShort( 0x3 ),  /* 3 */
/*  8 */  NdrFcShort( 0x30 ), /* ia64 Stack
size/offset = 48 */
/* 10 */  NdrFcShort( 0x0 ),  /* 0 */
/* 12 */  NdrFcShort( 0x8 ),  /* 8 */
/* 14 */  0x47,               /* Oi2 Flags:  srv must
size, clt must size, has return, has ext, */
                                0x3,                /*
3 */
/* 16 */  0xa,                /* 10 */
                                0x7,                /*
Ext Flags:  new corr desc, clt corr check, srv corr
check, */
/* 18 */  NdrFcShort( 0x20 ), /* 32 */
/* 20 */  NdrFcShort( 0x20 ), /* 32 */
/* 22 */  NdrFcShort( 0x0 ),  /* 0 */
/* 24 */  NdrFcShort( 0x0 ),  /* 0 */

          /* Parameter txn_in */

/* 26 */  NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 28 */  NdrFcShort( 0x8 ),  /* ia64 Stack
size/offset = 8 */
/* 30 */  NdrFcShort( 0x3ce ),          /* Type
Offset=974 */

          /* Parameter txn_out */

/* 32 */  NdrFcShort( 0x6113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=24 */
/* 34 */  NdrFcShort( 0x20 ), /* ia64 Stack
size/offset = 32 */
/* 36 */  NdrFcShort( 0x3e0 ),          /* Type
Offset=992 */

          /* Return value */

/* 38 */  NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 40 */  NdrFcShort( 0x28 ), /* ia64 Stack
size/offset = 40 */
/* 42 */  0x8,                /* FC_LONG */
                                0x0,                /*
0 */

          /* Procedure Payment */

/* 44 */  0x33,               /* FC_AUTO_HANDLE */
                                0x6c,               /*
Old Flags:  object, Oi2 */
/* 46 */  NdrFcLong( 0x0 ),   /* 0 */
/* 50 */  NdrFcShort( 0x4 ),  /* 4 */
/* 52 */  NdrFcShort( 0x30 ), /* ia64 Stack
size/offset = 48 */
/* 54 */  NdrFcShort( 0x0 ),  /* 0 */
/* 56 */  NdrFcShort( 0x8 ),  /* 8 */
/* 58 */  0x47,               /* Oi2 Flags:  srv must
size, clt must size, has return, has ext, */
                                0x3,                /*
3 */
/* 60 */  0xa,                /* 10 */
                                0x7,                /*
Ext Flags:  new corr desc, clt corr check, srv corr
check, */
/* 62 */  NdrFcShort( 0x20 ), /* 32 */
/* 64 */  NdrFcShort( 0x20 ), /* 32 */
/* 66 */  NdrFcShort( 0x0 ),  /* 0 */
/* 68 */  NdrFcShort( 0x0 ),  /* 0 */

          /* Parameter txn_in */

/* 70 */  NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 72 */  NdrFcShort( 0x8 ),  /* ia64 Stack
size/offset = 8 */
/* 74 */  NdrFcShort( 0x3ce ),          /* Type
Offset=974 */

          /* Parameter txn_out */

/* 76 */  NdrFcShort( 0x6113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=24 */
/* 78 */  NdrFcShort( 0x20 ), /* ia64 Stack
size/offset = 32 */
/* 80 */  NdrFcShort( 0x3e0 ),          /* Type
Offset=992 */

          /* Return value */

/* 82 */  NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 84 */  NdrFcShort( 0x28 ), /* ia64 Stack
size/offset = 40 */
/* 86 */  0x8,                /* FC_LONG */
                                0x0,                /*
0 */

          /* Procedure Delivery */

/* 88 */  0x33,               /* FC_AUTO_HANDLE */
                                0x6c,               /*
Old Flags:  object, Oi2 */
/* 90 */  NdrFcLong( 0x0 ),   /* 0 */
/* 94 */  NdrFcShort( 0x5 ),  /* 5 */
/* 96 */  NdrFcShort( 0x30 ), /* ia64 Stack
size/offset = 48 */
/* 98 */  NdrFcShort( 0x0 ),  /* 0 */
/* 100 */ NdrFcShort( 0x8 ),  /* 8 */
/* 102 */ 0x47,               /* Oi2 Flags:  srv must
size, clt must size, has return, has ext, */
                                0x3,                /*
3 */
/* 104 */ 0xa,                /* 10 */
```

```
                            0x7,                /*
Ext Flags:  new corr desc, clt corr check, srv corr
check, */
/* 106 */ NdrFcShort( 0x20 ), /* 32 */
/* 108 */ NdrFcShort( 0x20 ), /* 32 */
/* 110 */ NdrFcShort( 0x0 ),  /* 0 */
/* 112 */ NdrFcShort( 0x0 ),  /* 0 */

         /* Parameter txn_in */

/* 114 */ NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 116 */ NdrFcShort( 0x8 ),  /* ia64 Stack
size/offset = 8 */
/* 118 */ NdrFcShort( 0x3ce ),          /* Type
Offset=974 */

         /* Parameter txn_out */

/* 120 */ NdrFcShort( 0x6113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=24 */
/* 122 */ NdrFcShort( 0x20 ), /* ia64 Stack
size/offset = 32 */
/* 124 */ NdrFcShort( 0x3e0 ),          /* Type
Offset=992 */

         /* Return value */

/* 126 */ NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 128 */ NdrFcShort( 0x28 ), /* ia64 Stack
size/offset = 40 */
/* 130 */ 0x8,                /* FC_LONG */
                            0x0,                /*
0 */

         /* Procedure StockLevel */

/* 132 */ 0x33,               /* FC_AUTO_HANDLE */
                            0x6c,               /*
Old Flags:  object, Oi2 */
/* 134 */ NdrFcLong( 0x0 ),   /* 0 */
/* 138 */ NdrFcShort( 0x6 ),  /* 6 */
/* 140 */ NdrFcShort( 0x30 ), /* ia64 Stack
size/offset = 48 */
/* 142 */ NdrFcShort( 0x0 ),  /* 0 */
/* 144 */ NdrFcShort( 0x8 ),  /* 8 */
/* 146 */ 0x47,               /* Oi2 Flags:  srv must
size, clt must size, has return, has ext, */
                            0x3,                /*
3 */
/* 148 */ 0xa,                /* 10 */
                            0x7,                /*
Ext Flags:  new corr desc, clt corr check, srv corr
check, */
/* 150 */ NdrFcShort( 0x20 ), /* 32 */
/* 152 */ NdrFcShort( 0x20 ), /* 32 */
/* 154 */ NdrFcShort( 0x0 ),  /* 0 */
/* 156 */ NdrFcShort( 0x0 ),  /* 0 */

         /* Parameter txn_in */

/* 158 */ NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 160 */ NdrFcShort( 0x8 ),  /* ia64 Stack
size/offset = 8 */
/* 162 */ NdrFcShort( 0x3ce ),          /* Type
Offset=974 */

         /* Parameter txn_out */

/* 164 */ NdrFcShort( 0x6113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=24 */
/* 166 */ NdrFcShort( 0x20 ), /* ia64 Stack
size/offset = 32 */
/* 168 */ NdrFcShort( 0x3e0 ),          /* Type
Offset=992 */

         /* Return value */

/* 170 */ NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 172 */ NdrFcShort( 0x28 ), /* ia64 Stack
size/offset = 40 */
/* 174 */ 0x8,                /* FC_LONG */
                            0x0,                /*
0 */

         /* Procedure OrderStatus */

/* 176 */ 0x33,               /* FC_AUTO_HANDLE */
                            0x6c,               /*
Old Flags:  object, Oi2 */
/* 178 */ NdrFcLong( 0x0 ),   /* 0 */
/* 182 */ NdrFcShort( 0x7 ),  /* 7 */
/* 184 */ NdrFcShort( 0x30 ), /* ia64 Stack
size/offset = 48 */
/* 186 */ NdrFcShort( 0x0 ),  /* 0 */
/* 188 */ NdrFcShort( 0x8 ),  /* 8 */
/* 190 */ 0x47,               /* Oi2 Flags:  srv must
size, clt must size, has return, has ext, */
                            0x3,                /*
3 */
/* 192 */ 0xa,                /* 10 */
                            0x7,                /*
Ext Flags:  new corr desc, clt corr check, srv corr
check, */
/* 194 */ NdrFcShort( 0x20 ), /* 32 */
/* 196 */ NdrFcShort( 0x20 ), /* 32 */
/* 198 */ NdrFcShort( 0x0 ),  /* 0 */
/* 200 */ NdrFcShort( 0x0 ),  /* 0 */

         /* Parameter txn_in */

/* 202 */ NdrFcShort( 0x8b ), /* Flags:  must size,
must free, in, by val, */
/* 204 */ NdrFcShort( 0x8 ),  /* ia64 Stack
size/offset = 8 */
/* 206 */ NdrFcShort( 0x3ce ),          /* Type
Offset=974 */

         /* Parameter txn_out */

/* 208 */ NdrFcShort( 0x6113 ),         /* Flags:
must size, must free, out, simple ref, srv alloc
size=24 */
/* 210 */ NdrFcShort( 0x20 ), /* ia64 Stack
size/offset = 32 */
/* 212 */ NdrFcShort( 0x3e0 ),          /* Type
Offset=992 */

         /* Return value */

/* 214 */ NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 216 */ NdrFcShort( 0x28 ), /* ia64 Stack
size/offset = 40 */
/* 218 */ 0x8,                /* FC_LONG */
                            0x0,                /*
0 */

         /* Procedure CallSetComplete */

/* 220 */ 0x33,               /* FC_AUTO_HANDLE */
                            0x6c,               /*
Old Flags:  object, Oi2 */
/* 222 */ NdrFcLong( 0x0 ),   /* 0 */
/* 226 */ NdrFcShort( 0x8 ),  /* 8 */
/* 228 */ NdrFcShort( 0x10 ), /* ia64 Stack
size/offset = 16 */
/* 230 */ NdrFcShort( 0x0 ),  /* 0 */
/* 232 */ NdrFcShort( 0x8 ),  /* 8 */
/* 234 */ 0x44,               /* Oi2 Flags:  has
return, has ext, */
                            0x1,                /*
1 */
/* 236 */ 0xa,                /* 10 */
                            0x1,                /*
Ext Flags:  new corr desc, */
/* 238 */ NdrFcShort( 0x0 ),  /* 0 */
/* 240 */ NdrFcShort( 0x0 ),  /* 0 */
/* 242 */ NdrFcShort( 0x0 ),  /* 0 */
/* 244 */ NdrFcShort( 0x0 ),  /* 0 */

         /* Return value */

/* 246 */ NdrFcShort( 0x70 ), /* Flags:  out, return,
base type, */
/* 248 */ NdrFcShort( 0x8 ),  /* ia64 Stack
size/offset = 8 */
/* 250 */ 0x8,                /* FC_LONG */
                            0x0,                /*
0 */

                            0x0
        }
    };

static const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString =
    {
        0,
        {
                            NdrFcShort( 0x0 ),  /*
0 */
/*  2 */
```

```
                            0x12, 0x0,          /*
FC_UP */
/*  4 */  NdrFcShort( 0x3b6 ),          /* Offset=
950 (954) */
/*  6 */
                            0x2b,               /*
FC_NON_ENCAPSULATED_UNION */
                            0x9,                /*
FC_ULONG */
/*  8 */  0x7,                /* Corr desc: FC_USHORT
*/
                            0x0,                /*
*/
/* 10 */  NdrFcShort( 0xfff8 ),         /* -8 */
/* 12 */  NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 14 */  NdrFcShort( 0x2 ),  /* Offset= 2 (16) */
/* 16 */  NdrFcShort( 0x10 ), /* 16 */
/* 18 */  NdrFcShort( 0x2f ), /* 47 */
/* 20 */  NdrFcLong( 0x14 ),  /* 20 */
/* 24 */  NdrFcShort( 0x800b ),         /* Simple arm
type: FC_HYPER */
/* 26 */  NdrFcLong( 0x3 ),   /* 3 */
/* 30 */  NdrFcShort( 0x8008 ),         /* Simple arm
type: FC_LONG */
/* 32 */  NdrFcLong( 0x11 ),  /* 17 */
/* 36 */  NdrFcShort( 0x8001 ),         /* Simple arm
type: FC_BYTE */
/* 38 */  NdrFcLong( 0x2 ),   /* 2 */
/* 42 */  NdrFcShort( 0x8006 ),         /* Simple arm
type: FC_SHORT */
/* 44 */  NdrFcLong( 0x4 ),   /* 4 */
/* 48 */  NdrFcShort( 0x800a ),         /* Simple arm
type: FC_FLOAT */
/* 50 */  NdrFcLong( 0x5 ),   /* 5 */
/* 54 */  NdrFcShort( 0x800c ),         /* Simple arm
type: FC_DOUBLE */
/* 56 */  NdrFcLong( 0xb ),   /* 11 */
/* 60 */  NdrFcShort( 0x8006 ),         /* Simple arm
type: FC_SHORT */
/* 62 */  NdrFcLong( 0xa ),   /* 10 */
/* 66 */  NdrFcShort( 0x8008 ),         /* Simple arm
type: FC_LONG */
/* 68 */  NdrFcLong( 0x6 ),   /* 6 */
/* 72 */  NdrFcShort( 0xe8 ), /* Offset= 232 (304) */
/* 74 */  NdrFcLong( 0x7 ),   /* 7 */
/* 78 */  NdrFcShort( 0x800c ),         /* Simple arm
type: FC_DOUBLE */
/* 80 */  NdrFcLong( 0x8 ),   /* 8 */
/* 84 */  NdrFcShort( 0xe2 ), /* Offset= 226 (310) */
/* 86 */  NdrFcLong( 0xd ),   /* 13 */
/* 90 */  NdrFcShort( 0xf6 ), /* Offset= 246 (336) */
/* 92 */  NdrFcLong( 0x9 ),   /* 9 */
/* 96 */  NdrFcShort( 0x102 ),          /* Offset=
258 (354) */
/* 98 */  NdrFcLong( 0x2000 ),          /* 8192 */
/* 102 */ NdrFcShort( 0x10e ),         /* Offset=
270 (372) */
/* 104 */ NdrFcLong( 0x24 ),  /* 36 */
/* 108 */ NdrFcShort( 0x304 ),         /* Offset=
772 (880) */
/* 110 */ NdrFcLong( 0x4024 ),         /* 16420 */
/* 114 */ NdrFcShort( 0x2fe ),         /* Offset=
766 (880) */
/* 116 */ NdrFcLong( 0x4011 ),         /* 16401 */
/* 120 */ NdrFcShort( 0x2fc ),         /* Offset=
764 (884) */
/* 122 */ NdrFcLong( 0x4002 ),         /* 16386 */
/* 126 */ NdrFcShort( 0x2fa ),         /* Offset=
762 (888) */
/* 128 */ NdrFcLong( 0x4003 ),         /* 16387 */
/* 132 */ NdrFcShort( 0x2f8 ),         /* Offset=
760 (892) */
/* 134 */ NdrFcLong( 0x4014 ),         /* 16404 */
/* 138 */ NdrFcShort( 0x2f6 ),         /* Offset=
758 (896) */
/* 140 */ NdrFcLong( 0x4004 ),         /* 16388 */
/* 144 */ NdrFcShort( 0x2f4 ),         /* Offset=
756 (900) */
/* 146 */ NdrFcLong( 0x4005 ),         /* 16389 */
/* 150 */ NdrFcShort( 0x2f2 ),         /* Offset=
754 (904) */
/* 152 */ NdrFcLong( 0x400b ),         /* 16395 */
/* 156 */ NdrFcShort( 0x2dc ),         /* Offset=
732 (888) */
/* 158 */ NdrFcLong( 0x400a ),         /* 16394 */
/* 162 */ NdrFcShort( 0x2da ),         /* Offset=
730 (892) */
/* 164 */ NdrFcLong( 0x4006 ),         /* 16390 */
/* 168 */ NdrFcShort( 0x2e4 ),         /* Offset=
740 (908) */
/* 170 */ NdrFcLong( 0x4007 ),         /* 16391 */
/* 174 */ NdrFcShort( 0x2da ),         /* Offset=
730 (904) */
/* 176 */ NdrFcLong( 0x4008 ),         /* 16392 */
/* 180 */ NdrFcShort( 0x2dc ),         /* Offset=
732 (912) */
/* 182 */ NdrFcLong( 0x400d ),         /* 16397 */
/* 186 */ NdrFcShort( 0x2da ),         /* Offset=
730 (916) */
/* 188 */ NdrFcLong( 0x4009 ),         /* 16393 */
/* 192 */ NdrFcShort( 0x2d8 ),         /* Offset=
728 (920) */
/* 194 */ NdrFcLong( 0x6000 ),         /* 24576 */
/* 198 */ NdrFcShort( 0x2d6 ),         /* Offset=
726 (924) */
/* 200 */ NdrFcLong( 0x400c ),         /* 16396 */
/* 204 */ NdrFcShort( 0x2d4 ),         /* Offset=
724 (928) */
/* 206 */ NdrFcLong( 0x10 ),  /* 16 */
/* 210 */ NdrFcShort( 0x8002 ),        /* Simple arm
type: FC_CHAR */
/* 212 */ NdrFcLong( 0x12 ),  /* 18 */
/* 216 */ NdrFcShort( 0x8006 ),        /* Simple arm
type: FC_SHORT */
/* 218 */ NdrFcLong( 0x13 ),  /* 19 */
/* 222 */ NdrFcShort( 0x8008 ),        /* Simple arm
type: FC_LONG */
/* 224 */ NdrFcLong( 0x15 ),  /* 21 */
/* 228 */ NdrFcShort( 0x800b ),        /* Simple arm
type: FC_HYPER */
/* 230 */ NdrFcLong( 0x16 ),  /* 22 */
/* 234 */ NdrFcShort( 0x8008 ),        /* Simple arm
type: FC_LONG */
/* 236 */ NdrFcLong( 0x17 ),  /* 23 */
/* 240 */ NdrFcShort( 0x8008 ),        /* Simple arm
type: FC_LONG */
/* 242 */ NdrFcLong( 0xe ),   /* 14 */
/* 246 */ NdrFcShort( 0x2b2 ),         /* Offset=
690 (936) */
/* 248 */ NdrFcLong( 0x400e ),         /* 16398 */
/* 252 */ NdrFcShort( 0x2b6 ),         /* Offset=
694 (946) */
/* 254 */ NdrFcLong( 0x4010 ),         /* 16400 */
/* 258 */ NdrFcShort( 0x2b4 ),         /* Offset=
692 (950) */
/* 260 */ NdrFcLong( 0x4012 ),         /* 16402 */
/* 264 */ NdrFcShort( 0x270 ),         /* Offset=
624 (888) */
/* 266 */ NdrFcLong( 0x4013 ),         /* 16403 */
/* 270 */ NdrFcShort( 0x26e ),         /* Offset=
622 (892) */
/* 272 */ NdrFcLong( 0x4015 ),         /* 16405 */
/* 276 */ NdrFcShort( 0x26c ),         /* Offset=
620 (896) */
/* 278 */ NdrFcLong( 0x4016 ),         /* 16406 */
/* 282 */ NdrFcShort( 0x262 ),         /* Offset=
610 (892) */
/* 284 */ NdrFcLong( 0x4017 ),         /* 16407 */
/* 288 */ NdrFcShort( 0x25c ),         /* Offset=
604 (892) */
/* 290 */ NdrFcLong( 0x0 ),   /* 0 */
/* 294 */ NdrFcShort( 0x0 ),  /* Offset= 0 (294) */
/* 296 */ NdrFcLong( 0x1 ),   /* 1 */
/* 300 */ NdrFcShort( 0x0 ),  /* Offset= 0 (300) */
/* 302 */ NdrFcShort( 0xffff ),        /* Offset= -1
(301) */
/* 304 */
                            0x15,               /*
FC_STRUCT */
                            0x7,                /*
7 */
/* 306 */ NdrFcShort( 0x8 ),  /* 8 */
/* 308 */ 0xb,                /* FC_HYPER */
                            0x5b,               /*
FC_END */
/* 310 */
                            0x12, 0x0,          /*
FC_UP */
/* 312 */ NdrFcShort( 0xe ),  /* Offset= 14 (326) */
/* 314 */
                            0x1b,               /*
FC_CARRAY */
                            0x1,                /*
1 */
/* 316 */ NdrFcShort( 0x2 ),  /* 2 */
/* 318 */ 0x9,                /* Corr desc: FC_ULONG
*/
                            0x0,                /*
*/
/* 320 */ NdrFcShort( 0xfffc ),        /* -4 */
/* 322 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 324 */ 0x6,                /* FC_SHORT */
                            0x5b,               /*
FC_END */
/* 326 */
```

```
                            0x17,               /*
FC_CSTRUCT */
                            0x3,                /*
3 */
/* 328 */ NdrFcShort( 0x8 ),  /* 8 */
/* 330 */ NdrFcShort( 0xfff0 ),         /* Offset= -
16 (314) */
/* 332 */ 0x8,                /* FC_LONG */
                            0x8,                /*
FC_LONG */
/* 334 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 336 */
                            0x2f,               /*
FC_IP */
                            0x5a,               /*
FC_CONSTANT_IID */
/* 338 */ NdrFcLong( 0x0 ),   /* 0 */
/* 342 */ NdrFcShort( 0x0 ),  /* 0 */
/* 344 */ NdrFcShort( 0x0 ),  /* 0 */
/* 346 */ 0xc0,               /* 192 */
                            0x0,                /*
0 */
/* 348 */ 0x0,                /* 0 */
                            0x0,                /*
0 */
/* 350 */ 0x0,                /* 0 */
                            0x0,                /*
0 */
/* 352 */ 0x0,                /* 0 */
                            0x46,               /*
70 */
/* 354 */
                            0x2f,               /*
FC_IP */
                            0x5a,               /*
FC_CONSTANT_IID */
/* 356 */ NdrFcLong( 0x20400 ),         /* 132096 */
/* 360 */ NdrFcShort( 0x0 ),  /* 0 */
/* 362 */ NdrFcShort( 0x0 ),  /* 0 */
/* 364 */ 0xc0,               /* 192 */
                            0x0,                /*
0 */
/* 366 */ 0x0,                /* 0 */
                            0x0,                /*
0 */
/* 368 */ 0x0,                /* 0 */
                            0x0,                /*
0 */
/* 370 */ 0x0,                /* 0 */
                            0x46,               /*
70 */
/* 372 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 374 */ NdrFcShort( 0x2 ),  /* Offset= 2 (376) */
/* 376 */
                            0x12, 0x0,          /*
FC_UP */
/* 378 */ NdrFcShort( 0x1e4 ),          /* Offset=
484 (862) */
/* 380 */
                            0x2a,               /*
FC_ENCAPSULATED_UNION */
                            0x89,               /*
137 */
/* 382 */ NdrFcShort( 0x20 ), /* 32 */
/* 384 */ NdrFcShort( 0xa ),  /* 10 */
/* 386 */ NdrFcLong( 0x8 ),   /* 8 */
/* 390 */ NdrFcShort( 0x50 ), /* Offset= 80 (470) */
/* 392 */ NdrFcLong( 0xd ),   /* 13 */
/* 396 */ NdrFcShort( 0x70 ), /* Offset= 112 (508) */
/* 398 */ NdrFcLong( 0x9 ),   /* 9 */
/* 402 */ NdrFcShort( 0x90 ), /* Offset= 144 (546) */
/* 404 */ NdrFcLong( 0xc ),   /* 12 */
/* 408 */ NdrFcShort( 0xb0 ), /* Offset= 176 (584) */
/* 410 */ NdrFcLong( 0x24 ),  /* 36 */
/* 414 */ NdrFcShort( 0x102 ),          /* Offset=
258 (672) */
/* 416 */ NdrFcLong( 0x800d ),          /* 32781 */
/* 420 */ NdrFcShort( 0x11e ),          /* Offset=
286 (706) */
/* 422 */ NdrFcLong( 0x10 ),  /* 16 */
/* 426 */ NdrFcShort( 0x138 ),          /* Offset=
312 (738) */
/* 428 */ NdrFcLong( 0x2 ),   /* 2 */
/* 432 */ NdrFcShort( 0x14e ),          /* Offset=
334 (766) */
/* 434 */ NdrFcLong( 0x3 ),   /* 3 */
/* 438 */ NdrFcShort( 0x164 ),          /* Offset=
356 (794) */
/* 440 */ NdrFcLong( 0x14 ),  /* 20 */
/* 444 */ NdrFcShort( 0x17a ),          /* Offset=
378 (822) */
/* 446 */ NdrFcShort( 0xffff ),         /* Offset= -1
(445) */
/* 448 */
                            0x21,               /*
FC_BOGUS_ARRAY */
                            0x3,                /*
3 */
/* 450 */ NdrFcShort( 0x0 ),  /* 0 */
/* 452 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 454 */ NdrFcShort( 0x0 ),  /* 0 */
/* 456 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 458 */ NdrFcLong( 0xffffffff ),      /* -1 */
/* 462 */ NdrFcShort( 0x0 ),  /* Corr flags:  */
/* 464 */
                            0x12, 0x0,          /*
FC_UP */
/* 466 */ NdrFcShort( 0xff74 ),         /* Offset= -
140 (326) */
/* 468 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 470 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 472 */ NdrFcShort( 0x10 ), /* 16 */
/* 474 */ NdrFcShort( 0x0 ),  /* 0 */
/* 476 */ NdrFcShort( 0x6 ),  /* Offset= 6 (482) */
/* 478 */ 0x8,                /* FC_LONG */
                            0x40,               /*
FC_STRUCTPAD4 */
/* 480 */ 0x36,               /* FC_POINTER */
                            0x5b,               /*
FC_END */
/* 482 */
                            0x11, 0x0,          /*
FC_RP */
/* 484 */ NdrFcShort( 0xffdc ),         /* Offset= -
36 (448) */
/* 486 */
                            0x21,               /*
FC_BOGUS_ARRAY */
                            0x3,                /*
3 */
/* 488 */ NdrFcShort( 0x0 ),  /* 0 */
/* 490 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 492 */ NdrFcShort( 0x0 ),  /* 0 */
/* 494 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 496 */ NdrFcLong( 0xffffffff ),      /* -1 */
/* 500 */ NdrFcShort( 0x0 ),  /* Corr flags:  */
/* 502 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 504 */ NdrFcShort( 0xff58 ),         /* Offset= -
168 (336) */
/* 506 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 508 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 510 */ NdrFcShort( 0x10 ), /* 16 */
/* 512 */ NdrFcShort( 0x0 ),  /* 0 */
/* 514 */ NdrFcShort( 0x6 ),  /* Offset= 6 (520) */
/* 516 */ 0x8,                /* FC_LONG */
                            0x40,               /*
FC_STRUCTPAD4 */
/* 518 */ 0x36,               /* FC_POINTER */
                            0x5b,               /*
FC_END */
/* 520 */
                            0x11, 0x0,          /*
FC_RP */
/* 522 */ NdrFcShort( 0xffdc ),         /* Offset= -
36 (486) */
/* 524 */
                            0x21,               /*
FC_BOGUS_ARRAY */
                            0x3,                /*
3 */
/* 526 */ NdrFcShort( 0x0 ),  /* 0 */
```

```
/* 528 */ 0x19,              /* Corr desc:  field
pointer, FC_ULONG */
                             0x0,                /*
*/
/* 530 */ NdrFcShort( 0x0 ),  /* 0 */
/* 532 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 534 */ NdrFcLong( 0xffffffff ),      /* -1 */
/* 538 */ NdrFcShort( 0x0 ),  /* Corr flags:  */
/* 540 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                             0x0,                /*
0 */
/* 542 */ NdrFcShort( 0xff44 ),         /* Offset= -
188 (354) */
/* 544 */ 0x5c,               /* FC_PAD */
                             0x5b,               /*
FC_END */
/* 546 */
                             0x1a,               /*
FC_BOGUS_STRUCT */
                             0x3,                /*
3 */
/* 548 */ NdrFcShort( 0x10 ), /* 16 */
/* 550 */ NdrFcShort( 0x0 ),  /* 0 */
/* 552 */ NdrFcShort( 0x6 ),  /* Offset= 6 (558) */
/* 554 */ 0x8,                /* FC_LONG */
                             0x40,               /*
FC_STRUCTPAD4 */
/* 556 */ 0x36,               /* FC_POINTER */
                             0x5b,               /*
FC_END */
/* 558 */
                             0x11, 0x0,          /*
FC_RP */
/* 560 */ NdrFcShort( 0xffdc ),         /* Offset= -
36 (524) */
/* 562 */
                             0x21,               /*
FC_BOGUS_ARRAY */
                             0x3,                /*
3 */
/* 564 */ NdrFcShort( 0x0 ),  /* 0 */
/* 566 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                             0x0,                /*
*/
/* 568 */ NdrFcShort( 0x0 ),  /* 0 */
/* 570 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 572 */ NdrFcLong( 0xffffffff ),      /* -1 */
/* 576 */ NdrFcShort( 0x0 ),  /* Corr flags:  */
/* 578 */
                             0x12, 0x0,          /*
FC_UP */
/* 580 */ NdrFcShort( 0x176 ),          /* Offset=
374 (954) */
/* 582 */ 0x5c,               /* FC_PAD */
                             0x5b,               /*
FC_END */
/* 584 */
                             0x1a,               /*
FC_BOGUS_STRUCT */
                             0x3,                /*
3 */
/* 586 */ NdrFcShort( 0x10 ), /* 16 */
/* 588 */ NdrFcShort( 0x0 ),  /* 0 */
/* 590 */ NdrFcShort( 0x6 ),  /* Offset= 6 (596) */
/* 592 */ 0x8,                /* FC_LONG */
                             0x40,               /*
FC_STRUCTPAD4 */
/* 594 */ 0x36,               /* FC_POINTER */
                             0x5b,               /*
FC_END */
/* 596 */
                             0x11, 0x0,          /*
FC_RP */
/* 598 */ NdrFcShort( 0xffdc ),         /* Offset= -
36 (562) */
/* 600 */
                             0x2f,               /*
FC_IP */
                             0x5a,               /*
FC_CONSTANT_IID */
/* 602 */ NdrFcLong( 0x2f ),  /* 47 */
/* 606 */ NdrFcShort( 0x0 ),  /* 0 */
/* 608 */ NdrFcShort( 0x0 ),  /* 0 */
/* 610 */ 0xc0,               /* 192 */
                             0x0,                /*
0 */
/* 612 */ 0x0,                /* 0 */
                             0x0,                /*
0 */
/* 614 */ 0x0,                /* 0 */
                             0x0,                /*
0 */
/* 616 */ 0x0,                /* 0 */
                             0x46,               /*
70 */
/* 618 */
                             0x1b,               /*
FC_CARRAY */
                             0x0,                /*
0 */
/* 620 */ NdrFcShort( 0x1 ),  /* 1 */
/* 622 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                             0x0,                /*
*/
/* 624 */ NdrFcShort( 0x4 ),  /* 4 */
/* 626 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 628 */ 0x1,                /* FC_BYTE */
                             0x5b,               /*
FC_END */
/* 630 */
                             0x1a,               /*
FC_BOGUS_STRUCT */
                             0x3,                /*
3 */
/* 632 */ NdrFcShort( 0x18 ), /* 24 */
/* 634 */ NdrFcShort( 0x0 ),  /* 0 */
/* 636 */ NdrFcShort( 0xa ),  /* Offset= 10 (646) */
/* 638 */ 0x8,                /* FC_LONG */
                             0x8,                /*
FC_LONG */
/* 640 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                             0x0,                /*
0 */
/* 642 */ NdrFcShort( 0xffd6 ),         /* Offset= -
42 (600) */
/* 644 */ 0x36,               /* FC_POINTER */
                             0x5b,               /*
FC_END */
/* 646 */
                             0x12, 0x0,          /*
FC_UP */
/* 648 */ NdrFcShort( 0xffe2 ),         /* Offset= -
30 (618) */
/* 650 */
                             0x21,               /*
FC_BOGUS_ARRAY */
                             0x3,                /*
3 */
/* 652 */ NdrFcShort( 0x0 ),  /* 0 */
/* 654 */ 0x19,               /* Corr desc:  field
pointer, FC_ULONG */
                             0x0,                /*
*/
/* 656 */ NdrFcShort( 0x0 ),  /* 0 */
/* 658 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 660 */ NdrFcLong( 0xffffffff ),      /* -1 */
/* 664 */ NdrFcShort( 0x0 ),  /* Corr flags:  */
/* 666 */
                             0x12, 0x0,          /*
FC_UP */
/* 668 */ NdrFcShort( 0xffda ),         /* Offset= -
38 (630) */
/* 670 */ 0x5c,               /* FC_PAD */
                             0x5b,               /*
FC_END */
/* 672 */
                             0x1a,               /*
FC_BOGUS_STRUCT */
                             0x3,                /*
3 */
/* 674 */ NdrFcShort( 0x10 ), /* 16 */
/* 676 */ NdrFcShort( 0x0 ),  /* 0 */
/* 678 */ NdrFcShort( 0x6 ),  /* Offset= 6 (684) */
/* 680 */ 0x8,                /* FC_LONG */
                             0x40,               /*
FC_STRUCTPAD4 */
/* 682 */ 0x36,               /* FC_POINTER */
                             0x5b,               /*
FC_END */
/* 684 */
                             0x11, 0x0,          /*
FC_RP */
/* 686 */ NdrFcShort( 0xffdc ),         /* Offset= -
36 (650) */
/* 688 */
                             0x1d,               /*
FC_SMFARRAY */
                             0x0,                /*
0 */
/* 690 */ NdrFcShort( 0x8 ),  /* 8 */
/* 692 */ 0x1,                /* FC_BYTE */
```

```
                            0x5b,               /*
FC_END */
/* 694 */
                            0x15,               /*
FC_STRUCT */
                            0x3,                /*
3 */
/* 696 */ NdrFcShort( 0x10 ), /* 16 */
/* 698 */ 0x8,              /* FC_LONG */
                            0x6,                /*
FC_SHORT */
/* 700 */ 0x6,              /* FC_SHORT */
                            0x4c,               /*
FC_EMBEDDED_COMPLEX */
/* 702 */ 0x0,              /* 0 */
                            NdrFcShort( 0xfff1 ),
          /* Offset= -15 (688) */
                            0x5b,               /*
FC_END */
/* 706 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 708 */ NdrFcShort( 0x20 ), /* 32 */
/* 710 */ NdrFcShort( 0x0 ),  /* 0 */
/* 712 */ NdrFcShort( 0xa ),  /* Offset= 10 (722) */
/* 714 */ 0x8,              /* FC_LONG */
                            0x40,               /*
FC_STRUCTPAD4 */
/* 716 */ 0x36,             /* FC_POINTER */
                            0x4c,               /*
FC_EMBEDDED_COMPLEX */
/* 718 */ 0x0,              /* 0 */
                            NdrFcShort( 0xffe7 ),
          /* Offset= -25 (694) */
                            0x5b,               /*
FC_END */
/* 722 */
                            0x11, 0x0,          /*
FC_RP */
/* 724 */ NdrFcShort( 0xff12 ),         /* Offset= -
238 (486) */
/* 726 */
                            0x1b,               /*
FC_CARRAY */
                            0x0,                /*
0 */
/* 728 */ NdrFcShort( 0x1 ),  /* 1 */
/* 730 */ 0x19,             /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 732 */ NdrFcShort( 0x0 ),  /* 0 */
/* 734 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 736 */ 0x1,              /* FC_BYTE */
                            0x5b,               /*
FC_END */
/* 738 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 740 */ NdrFcShort( 0x10 ), /* 16 */
/* 742 */ NdrFcShort( 0x0 ),  /* 0 */
/* 744 */ NdrFcShort( 0x6 ),  /* Offset= 6 (750) */
/* 746 */ 0x8,              /* FC_LONG */
                            0x40,               /*
FC_STRUCTPAD4 */
/* 748 */ 0x36,             /* FC_POINTER */
                            0x5b,               /*
FC_END */
/* 750 */
                            0x12, 0x0,          /*
FC_UP */
/* 752 */ NdrFcShort( 0xffe6 ),         /* Offset= -
26 (726) */
/* 754 */
                            0x1b,               /*
FC_CARRAY */
                            0x1,                /*
1 */
/* 756 */ NdrFcShort( 0x2 ),  /* 2 */
/* 758 */ 0x19,             /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 760 */ NdrFcShort( 0x0 ),  /* 0 */
/* 762 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 764 */ 0x6,              /* FC_SHORT */
                            0x5b,               /*
FC_END */
/* 766 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 768 */ NdrFcShort( 0x10 ), /* 16 */
/* 770 */ NdrFcShort( 0x0 ),  /* 0 */
/* 772 */ NdrFcShort( 0x6 ),  /* Offset= 6 (778) */
/* 774 */ 0x8,              /* FC_LONG */
                            0x40,               /*
FC_STRUCTPAD4 */
/* 776 */ 0x36,             /* FC_POINTER */
                            0x5b,               /*
FC_END */
/* 778 */
                            0x12, 0x0,          /*
FC_UP */
/* 780 */ NdrFcShort( 0xffe6 ),         /* Offset= -
26 (754) */
/* 782 */
                            0x1b,               /*
FC_CARRAY */
                            0x3,                /*
3 */
/* 784 */ NdrFcShort( 0x4 ),  /* 4 */
/* 786 */ 0x19,             /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 788 */ NdrFcShort( 0x0 ),  /* 0 */
/* 790 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 792 */ 0x8,              /* FC_LONG */
                            0x5b,               /*
FC_END */
/* 794 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 796 */ NdrFcShort( 0x10 ), /* 16 */
/* 798 */ NdrFcShort( 0x0 ),  /* 0 */
/* 800 */ NdrFcShort( 0x6 ),  /* Offset= 6 (806) */
/* 802 */ 0x8,              /* FC_LONG */
                            0x40,               /*
FC_STRUCTPAD4 */
/* 804 */ 0x36,             /* FC_POINTER */
                            0x5b,               /*
FC_END */
/* 806 */
                            0x12, 0x0,          /*
FC_UP */
/* 808 */ NdrFcShort( 0xffe6 ),         /* Offset= -
26 (782) */
/* 810 */
                            0x1b,               /*
FC_CARRAY */
                            0x7,                /*
7 */
/* 812 */ NdrFcShort( 0x8 ),  /* 8 */
/* 814 */ 0x19,             /* Corr desc:  field
pointer, FC_ULONG */
                            0x0,                /*
*/
/* 816 */ NdrFcShort( 0x0 ),  /* 0 */
/* 818 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 820 */ 0xb,              /* FC_HYPER */
                            0x5b,               /*
FC_END */
/* 822 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 824 */ NdrFcShort( 0x10 ), /* 16 */
/* 826 */ NdrFcShort( 0x0 ),  /* 0 */
/* 828 */ NdrFcShort( 0x6 ),  /* Offset= 6 (834) */
/* 830 */ 0x8,              /* FC_LONG */
                            0x40,               /*
FC_STRUCTPAD4 */
/* 832 */ 0x36,             /* FC_POINTER */
                            0x5b,               /*
FC_END */
/* 834 */
                            0x12, 0x0,          /*
FC_UP */
/* 836 */ NdrFcShort( 0xffe6 ),         /* Offset= -
26 (810) */
/* 838 */
                            0x15,               /*
FC_STRUCT */
```

```
                            0x3,                /*
3 */
/* 840 */ NdrFcShort( 0x8 ),  /* 8 */
/* 842 */ 0x8,                /* FC_LONG */
                            0x8,                /*
FC_LONG */
/* 844 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 846 */
                            0x1b,               /*
FC_CARRAY */
                            0x3,                /*
3 */
/* 848 */ NdrFcShort( 0x8 ),  /* 8 */
/* 850 */ 0x7,                /* Corr desc: FC_USHORT
*/
                            0x0,                /*
*/
/* 852 */ NdrFcShort( 0xffc8 ),         /* -56 */
/* 854 */ NdrFcShort( 0x1 ),  /* Corr flags:  early,
*/
/* 856 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 858 */ NdrFcShort( 0xffec ),         /* Offset= -
20 (838) */
/* 860 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 862 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x3,                /*
3 */
/* 864 */ NdrFcShort( 0x38 ), /* 56 */
/* 866 */ NdrFcShort( 0xffec ),         /* Offset= -
20 (846) */
/* 868 */ NdrFcShort( 0x0 ),  /* Offset= 0 (868) */
/* 870 */ 0x6,                /* FC_SHORT */
                            0x6,                /*
FC_SHORT */
/* 872 */ 0x8,                /* FC_LONG */
                            0x8,                /*
FC_LONG */
/* 874 */ 0x40,               /* FC_STRUCTPAD4 */
                            0x4c,               /*
FC_EMBEDDED_COMPLEX */
/* 876 */ 0x0,                /* 0 */
                            NdrFcShort( 0xfe0f ),
          /* Offset= -497 (380) */
                            0x5b,               /*
FC_END */
/* 880 */
                            0x12, 0x0,          /*
FC_UP */
/* 882 */ NdrFcShort( 0xff04 ),         /* Offset= -
252 (630) */
/* 884 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 886 */ 0x1,                /* FC_BYTE */
                            0x5c,               /*
FC_PAD */
/* 888 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 890 */ 0x6,                /* FC_SHORT */
                            0x5c,               /*
FC_PAD */
/* 892 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 894 */ 0x8,                /* FC_LONG */
                            0x5c,               /*
FC_PAD */
/* 896 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 898 */ 0xb,                /* FC_HYPER */
                            0x5c,               /*
FC_PAD */
/* 900 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 902 */ 0xa,                /* FC_FLOAT */
                            0x5c,               /*
FC_PAD */
/* 904 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 906 */ 0xc,                /* FC_DOUBLE */
                            0x5c,               /*
FC_PAD */
/* 908 */
                            0x12, 0x0,          /*
FC_UP */
/* 910 */ NdrFcShort( 0xfda2 ),         /* Offset= -
606 (304) */
/* 912 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 914 */ NdrFcShort( 0xfda4 ),         /* Offset= -
604 (310) */
/* 916 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 918 */ NdrFcShort( 0xfdba ),         /* Offset= -
582 (336) */
/* 920 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 922 */ NdrFcShort( 0xfdc8 ),         /* Offset= -
568 (354) */
/* 924 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 926 */ NdrFcShort( 0xfdd6 ),         /* Offset= -
554 (372) */
/* 928 */
                            0x12, 0x10,         /*
FC_UP [pointer_deref] */
/* 930 */ NdrFcShort( 0x2 ),  /* Offset= 2 (932) */
/* 932 */
                            0x12, 0x0,          /*
FC_UP */
/* 934 */ NdrFcShort( 0x14 ), /* Offset= 20 (954) */
/* 936 */
                            0x15,               /*
FC_STRUCT */
                            0x7,                /*
7 */
/* 938 */ NdrFcShort( 0x10 ), /* 16 */
/* 940 */ 0x6,                /* FC_SHORT */
                            0x1,                /*
FC_BYTE */
/* 942 */ 0x1,                /* FC_BYTE */
                            0x8,                /*
FC_LONG */
/* 944 */ 0xb,                /* FC_HYPER */
                            0x5b,               /*
FC_END */
/* 946 */
                            0x12, 0x0,          /*
FC_UP */
/* 948 */ NdrFcShort( 0xfff4 ),         /* Offset= -
12 (936) */
/* 950 */
                            0x12, 0x8,          /*
FC_UP [simple_pointer] */
/* 952 */ 0x2,                /* FC_CHAR */
                            0x5c,               /*
FC_PAD */
/* 954 */
                            0x1a,               /*
FC_BOGUS_STRUCT */
                            0x7,                /*
7 */
/* 956 */ NdrFcShort( 0x20 ), /* 32 */
/* 958 */ NdrFcShort( 0x0 ),  /* 0 */
/* 960 */ NdrFcShort( 0x0 ),  /* Offset= 0 (960) */
/* 962 */ 0x8,                /* FC_LONG */
                            0x8,                /*
FC_LONG */
/* 964 */ 0x6,                /* FC_SHORT */
                            0x6,                /*
FC_SHORT */
/* 966 */ 0x6,                /* FC_SHORT */
                            0x6,                /*
FC_SHORT */
/* 968 */ 0x4c,               /* FC_EMBEDDED_COMPLEX
*/
                            0x0,                /*
0 */
/* 970 */ NdrFcShort( 0xfc3c ),         /* Offset= -
964 (6) */
/* 972 */ 0x5c,               /* FC_PAD */
                            0x5b,               /*
FC_END */
/* 974 */ 0xb4,               /* FC_USER_MARSHAL */
                            0x83,               /*
131 */
/* 976 */ NdrFcShort( 0x0 ),  /* 0 */
/* 978 */ NdrFcShort( 0x18 ), /* 24 */
/* 980 */ NdrFcShort( 0x0 ),  /* 0 */
/* 982 */ NdrFcShort( 0xfc2c ),         /* Offset= -
980 (2) */
```

```
/* 984 */
                            0x11, 0x4,          /*
FC_RP [alloced_on_stack] */
/* 986 */ NdrFcShort( 0x6 ),  /* Offset= 6 (992) */
/* 988 */
                            0x13, 0x0,          /*
FC_OP */
/* 990 */ NdrFcShort( 0xffdc ),         /* Offset= -
36 (954) */
/* 992 */ 0xb4,               /* FC_USER_MARSHAL */
                            0x83,               /*
131 */
/* 994 */ NdrFcShort( 0x0 ),  /* 0 */
/* 996 */ NdrFcShort( 0x18 ), /* 24 */
/* 998 */ NdrFcShort( 0x0 ),  /* 0 */
/* 1000 */          NdrFcShort( 0xfff4 ),         /*
Offset= -12 (988) */

                            0x0
        }
    };

static const USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[ WIRE_MARSHAL_TABLE_SIZE ] =
        {

            {
            VARIANT_UserSize
            ,VARIANT_UserMarshal
            ,VARIANT_UserUnmarshal
            ,VARIANT_UserFree
            }

        };



/* Standard interface: __MIDL_itf_tpcc_com_ps_0000,
ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0x00,0x00,0x00,0x00,0
x00,0x00,0x00,0x00}} */


/* Object interface: IUnknown, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0
x00,0x00,0x00,0x46}} */


/* Object interface: ITPCC, ver. 0.0,

GUID={0xFEEE6AA2,0x84B1,0x11d2,{0xBA,0x47,0x00,0xC0,0
x4F,0xBF,0xE0,0x8B}} */

#pragma code_seg(".orpc")
static const unsigned short
ITPCC_FormatStringOffsetTable[] =
    {
    0,
    44,
    88,
    132,
    176,
    220
    };

static const MIDL_STUBLESS_PROXY_INFO ITPCC_ProxyInfo
=
    {
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &ITPCC_FormatStringOffsetTable[-3],
    0,
    0,
    0
    };


static const MIDL_SERVER_INFO ITPCC_ServerInfo =
    {
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &ITPCC_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0};
CINTERFACE_PROXY_VTABLE(9) _ITPCCProxyVtbl =
{
    &ITPCC_ProxyInfo,
    &IID_ITPCC,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    (void *) (INT_PTR) -1 /* ITPCC::NewOrder */ ,
    (void *) (INT_PTR) -1 /* ITPCC::Payment */ ,
    (void *) (INT_PTR) -1 /* ITPCC::Delivery */ ,
    (void *) (INT_PTR) -1 /* ITPCC::StockLevel */ ,
    (void *) (INT_PTR) -1 /* ITPCC::OrderStatus */ ,
    (void *) (INT_PTR) -1 /* ITPCC::CallSetComplete
*/
};

const CInterfaceStubVtbl _ITPCCStubVtbl =
{
    &IID_ITPCC,
    &ITPCC_ServerInfo,
    9,
    0, /* pure interpreted */
    CStdStubBuffer_METHODS
};

static const MIDL_STUB_DESC Object_StubDesc =
    {
    0,
    NdrOleAllocate,
    NdrOleFree,
    0,
    0,
    0,
    0,
    0,
    __MIDL_TypeFormatString.Format,
    1, /* -error bounds_check flag */
    0x50002, /* Ndr library version */
    0,
    0x6000169, /* MIDL Version 6.0.361 */
    0,
    UserMarshalRoutines,
    0,  /* notify & notify_flag routine table */
    0x1, /* MIDL flag */
    0, /* cs routines */
    0,   /* proxy/server info */
    0   /* Reserved5 */
    };

const CInterfaceProxyVtbl *
_tpcc_com_ps_ProxyVtblList[] =
{
    ( CInterfaceProxyVtbl *) &_ITPCCProxyVtbl,
    0
};

const CInterfaceStubVtbl *
_tpcc_com_ps_StubVtblList[] =
{
    ( CInterfaceStubVtbl *) &_ITPCCStubVtbl,
    0
};

PCInterfaceName const
_tpcc_com_ps_InterfaceNamesList[] =
{
    "ITPCC",
    0
};


#define _tpcc_com_ps_CHECK_IID(n)
          IID_GENERIC_CHECK_IID( _tpcc_com_ps, pIID,
n)

int __stdcall _tpcc_com_ps_IID_Lookup( const IID *
pIID, int * pIndex )
{

    if(!_tpcc_com_ps_CHECK_IID(0))
        {
        *pIndex = 0;
        return 1;
        }

    return 0;
}

const ExtendedProxyFileInfo tpcc_com_ps_ProxyFileInfo
=
{
    (PCInterfaceProxyVtblList *) &
_tpcc_com_ps_ProxyVtblList,
    (PCInterfaceStubVtblList *) &
_tpcc_com_ps_StubVtblList,
    (const PCInterfaceName * ) &
_tpcc_com_ps_InterfaceNamesList,
    0, // no delegation
    & _tpcc_com_ps_IID_Lookup,
    1,
```

```
    2,
    0, /* table of [async_uuid] interfaces */
    0, /* Filler1 */
    0, /* Filler2 */
    0  /* Filler3 */
};
#if _MSC_VER >= 1200
#pragma warning(pop)
#endif


#endif /* defined(_M_IA64) || defined(_M_AMD64)*/
```

tpcc_odbc.cpp

```
/*        FILE:               TPCC_ODBC.CPP
 *                                  Microsoft
TPC-C Kit Ver. 4.69.000
 *                                  Copyright
Microsoft, 2002
 *              All Rights Reserved
 *
 *                                  Version
4.10.000 audited by Richard Gimarc, Performance
Metrics, 3/17/99
 *
 *        PURPOSE:  Implements ODBC calls for TPC-C
txns.
 *        Contact:  Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *                4.42.000 - changed w_id fields
from short to long to support >32K warehouses
 *                4.20.000 - updated rev number to
match kit
 *                4.10.001 - not deleting error
class in catch handler on deadlock retry;
 *                                          not a
functional bug, but a memory leak
 *                4.69.000 - updated rev number to
match kit
 */

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

#define DBNTWIN32
#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>

//#define COMPILE_FOR_SNAC    // define that to
compile for SQL Native Client; comment out to use
MDAC

#ifndef COMPILE_FOR_SNAC
#include <odbcss.h>
#else
// Compile for SNAC
#include <sqlncli.h>
#endif

#ifdef ICECAP
#include <icapexp.h>
#endif

// need to declare functions for export
#define DllDecl  __declspec( dllexport )

#include "..\..\common\src\error.h"
#include "..\..\common\src\trans.h"
#include "..\..\common\src\txn_base.h"
#include "tpcc_odbc.h"

// version string; must match return value from
tpcc_version stored proc
const char          sVersion[] = "4.20.000";

const iMaxRetries = 3;                  // how many
retries on deadlock
//const iMaxRetries = 0;                // for
debugging

const int iErrOleDbProvider = 7312;
const char sErrTimeoutExpired[] = "Timeout expired";

static SQLHENV henv = SQL_NULL_HENV;
                    // ODBC environment handle

BOOL APIENTRY DllMain(HMODULE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved)
{
          switch( ul_reason_for_call )
          {
                    case DLL_PROCESS_ATTACH:

          DisableThreadLibraryCalls(hModule);
                              if (
SQLAllocHandleStd(SQL_HANDLE_ENV, SQL_NULL_HANDLE,
&henv) != SQL_SUCCESS )
                                        return FALSE;
                              break;

                    case DLL_PROCESS_DETACH:
                              if (henv != NULL)

          SQLFreeEnv(henv);
                              break;

                    default:
                              /* nothing */;
          }
          return TRUE;
}

/* FUNCTION: CTPCC_ODBC_ERR::ErrorText
 *
 */

char* CTPCC_ODBC_ERR::ErrorText(void)
{
          int i;

          static SERRORMSG errorMsgs[] =
          {
                    { ERR_WRONG_SP_VERSION,
          "Wrong version of stored procs on database
server"   },
                    { ERR_INVALID_CUST,
          "Invalid Customer id,name."
                                                  },
                    { ERR_NO_SUCH_ORDER,
          "No orders found for customer."
                                                  },
                    { ERR_RETRIED_TRANS,
          "Retries before transaction succeeded."
                              },
                    { ERR_INVALID_NEW_ORDER_PARAM,
          "New Order parameter invalid."
                                        },
                    { 0,
                              ""



          }
          };

          static char szNotFound[] = "Unknown error
number.";

          for(i=0; errorMsgs[i].szMsg[0]; i++)
          {
                    if ( m_errno ==
errorMsgs[i].iError )
                              break;
          }
          if ( !errorMsgs[i].szMsg[0] )
                    return szNotFound;
          else
                    return errorMsgs[i].szMsg;
}

// wrapper routine for class constructor
__declspec(dllexport) CTPCC_ODBC* CTPCC_ODBC_new(
          LPCSTR szServer,              // name of
SQL server
          LPCSTR szUser,                          //
user name for login
          LPCSTR szPassword,            // password
for login
          LPCSTR szHost,                          //
not used
          LPCSTR szDatabase,            // name of
database to use
          LPCWSTR szSPPrefix,           // prefix to
append to the stored procedure names
          BOOL bCallNoDuplicatesNewOrder ) // whether
to check for non-duplicate items in NewOrder and call
a new SP
{
          return new CTPCC_ODBC( szServer, szUser,
szPassword, szHost, szDatabase, szSPPrefix,
bCallNoDuplicatesNewOrder );
```

```
}


CTPCC_ODBC::CTPCC_ODBC (
          LPCSTR szServer,
          // name of SQL server
          LPCSTR szUser,
                    // user name for login
          LPCSTR szPassword,
          // password for login
          LPCSTR szHost,
                    // not used
          LPCSTR szDatabase,
          // name of database to use
          LPCWSTR szSPPrefix,
          // prefix to append to the stored procedure
names
          BOOL      bCallNoDuplicatesNewOrder //
whether to check for non-duplicate items in NewOrder
and call a new SP
          )
:
m_bCallNoDuplicatesNewOrder(bCallNoDuplicatesNewOrder
)
{
          RETCODE             rc;

          // initialization
          m_hdbc = SQL_NULL_HDBC;
          m_hstmt = SQL_NULL_HSTMT;

          m_hstmtNewOrder = SQL_NULL_HSTMT;
          m_hstmtPayment = SQL_NULL_HSTMT;
          m_hstmtDelivery = SQL_NULL_HSTMT;
          m_hstmtOrderStatus = SQL_NULL_HSTMT;
          m_hstmtStockLevel = SQL_NULL_HSTMT;

          m_descNewOrderCols1 = SQL_NULL_HDESC;
          m_descNewOrderCols2 = SQL_NULL_HDESC;
          m_descOrderStatusCols1 = SQL_NULL_HDESC;
          m_descOrderStatusCols2 = SQL_NULL_HDESC;

          wcsncpy(m_szSPPrefix, szSPPrefix,
sizeof(m_szSPPrefix)/sizeof(m_szSPPrefix[0]));


          if ( SQLAllocHandle(SQL_HANDLE_DBC, henv,
&m_hdbc) != SQL_SUCCESS )

          ThrowError(CODBCERR::eAllocHandle);

          if ( SQLSetConnectOption(m_hdbc,
SQL_PACKET_SIZE, 4096) != SQL_SUCCESS )

          ThrowError(CODBCERR::eConnOption);

          {
                    char
          szConnectStr[256];
                    char
          szOutStr[1024];
                    SQLSMALLINT
          iOutStrLen;

#ifndef COMPILE_FOR_SNAC
                    sprintf( szConnectStr,
"DRIVER=SQL
Server;SERVER=%s;UID=%s;PWD=%s;DATABASE=%s",
                              szServer, szUser,
szPassword, szDatabase );
#else
                    // Compile for SNAC
                    sprintf( szConnectStr,
"DRIVER=SQL Native
Client;SERVER=%s;UID=%s;PWD=%s;DATABASE=%s",
                              szServer, szUser,
szPassword, szDatabase );
#endif
                    rc = SQLDriverConnect(m_hdbc,
NULL, (SQLCHAR*)szConnectStr, sizeof(szConnectStr),
                              (SQLCHAR*)szOutStr,
sizeof(szOutStr), &iOutStrLen, SQL_DRIVER_NOPROMPT );

                    if (rc != SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)

          ThrowError(CODBCERR::eConnect);
          }

          if (SQLAllocHandle(SQL_HANDLE_STMT, m_hdbc,
&m_hstmt) != SQL_SUCCESS)

          ThrowError(CODBCERR::eAllocHandle);

          {
                    char                buffer[128];

                    // set some options affecting
connection behavior
                    strcpy(buffer, "set nocount on
set XACT_ABORT ON");
                    rc = SQLExecDirect(m_hstmt,
(unsigned char *)buffer, SQL_NTS);
                    if (rc != SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)

          ThrowError(CODBCERR::eExecDirect);

                    // verify that version of stored
procs on server is correct
                    char db_sp_version[10];
                    strcpy(buffer, "{call
tpcc_version}");
                    rc = SQLExecDirect(m_hstmt,
(unsigned char *)buffer, SQL_NTS);
                    if (rc != SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)

          ThrowError(CODBCERR::eExecDirect);
                    if ( SQLBindCol(m_hstmt, 1,
SQL_C_CHAR, &db_sp_version, sizeof(db_sp_version),
NULL) != SQL_SUCCESS )

          ThrowError(CODBCERR::eBindCol);
                    if ( SQLFetch(m_hstmt) ==
SQL_ERROR )

          ThrowError(CODBCERR::eFetch);
                    if
(strcmp(db_sp_version,sVersion))
                              throw new
CTPCC_ODBC_ERR( CTPCC_ODBC_ERR::ERR_WRONG_SP_VERSION
);

                    SQLFreeHandle(SQL_HANDLE_STMT,
m_hstmt);
          }

          // Bind parameters for each of the
transactions
          InitNewOrderParams();
          InitPaymentParams();
          InitOrderStatusParams();
          InitDeliveryParams();
          InitStockLevelParams();
}

CTPCC_ODBC::~CTPCC_ODBC( void )
{
          // note: descriptors are automatically
released when the connection is dropped
          SQLFreeHandle(SQL_HANDLE_STMT,
m_hstmtNewOrder);
          SQLFreeHandle(SQL_HANDLE_STMT,
m_hstmtPayment);
          SQLFreeHandle(SQL_HANDLE_STMT,
m_hstmtDelivery);
          SQLFreeHandle(SQL_HANDLE_STMT,
m_hstmtOrderStatus);
          SQLFreeHandle(SQL_HANDLE_STMT,
m_hstmtStockLevel);

          SQLDisconnect(m_hdbc);
          SQLFreeHandle(SQL_HANDLE_DBC, m_hdbc);
}

//void CTPCC_ODBC::ThrowError( CODBCERR::ACTION
eAction )
void CTPCC_ODBC::ThrowError( RETCODE eAction )
{
          RETCODE             rc;
          SDWORD              lNativeError;
          char                szState[6];
          char
          szMsg[SQL_MAX_MESSAGE_LENGTH];
          char
          szTmp[6*SQL_MAX_MESSAGE_LENGTH];
          CODBCERR   *pODBCErr;
          // not allocated until needed (maybe never)

          pODBCErr = new CODBCERR();

          pODBCErr->m_NativeError = 0;
          //pODBCErr->m_eAction = eAction;
          pODBCErr->m_eAction =
(CODBCERR::ACTION)eAction;
          pODBCErr->m_bDeadLock = FALSE;

          szTmp[0] = 0;
```

```
        szMsg[0] = 0;
        while (TRUE)
        {
                rc = SQLError(henv, m_hdbc,
m_hstmt, (BYTE *)&szState, &lNativeError,

(BYTE *)&szMsg, sizeof(szMsg), NULL);
                if (rc == SQL_NO_DATA)
                {
                        break;
                }

                if (rc != SQL_SUCCESS)
                {
                        break;
                }

                // check for deadlock
                if (lNativeError == 1205 ||
(lNativeError == iErrOleDbProvider &&
                        strstr(szMsg,
sErrTimeoutExpired) != NULL))
                        pODBCErr->m_bDeadLock =
TRUE;

                // capture the (first) database
error
                if (pODBCErr->m_NativeError == 0
&& lNativeError != 0)
                        pODBCErr->m_NativeError
= lNativeError;

                // quit if there isn't enough
room to concatenate error text
                if ( (strlen(szMsg) + 2) >
(sizeof(szTmp) - strlen(szTmp)) )
                        break;

                // include line break after first
error msg
                if (szTmp[0] != 0)
                        strcat( szTmp, "\n");
                strcat( szTmp, szMsg );
        }

        if (pODBCErr->m_odbcerrstr != NULL)
        {
                delete [] pODBCErr->m_odbcerrstr;
                pODBCErr->m_odbcerrstr = NULL;
        }

        if (strlen(szTmp) > 0)
        {
                pODBCErr->m_odbcerrstr = new
char[ strlen(szTmp)+1 ];
                strcpy( pODBCErr->m_odbcerrstr,
szTmp );
        }

        SQLFreeStmt(m_hstmt, SQL_CLOSE);
        throw pODBCErr;
}
```

```
void CTPCC_ODBC::InitStockLevelParams()
{
        if (   SQLAllocHandle(SQL_HANDLE_STMT,
m_hdbc, &m_hstmtStockLevel) != SQL_SUCCESS )

        ThrowError(CODBCERR::eAllocHandle);

        m_hstmt = m_hstmtStockLevel;

        int i = 0;
        if (   SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.StockLevel.w_id, 0, NULL) != SQL_SUCCESS
                || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.StockLevel.d_id, 0, NULL) != SQL_SUCCESS
                || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SSHORT, SQL_SMALLINT, 0, 0,
&m_txn.StockLevel.threshold, 0, NULL) != SQL_SUCCESS
                )
                ThrowError(CODBCERR::eBindParam);

        if ( SQLBindCol(m_hstmt, 1, SQL_C_SLONG,
&m_txn.StockLevel.low_stock, 0, NULL) != SQL_SUCCESS
)
                ThrowError(CODBCERR::eBindCol);

        //Compose Stock Level statement
        _snwprintf(m_szStockLevelCommand,
sizeof(m_szStockLevelCommand)/sizeof(m_szStockLevelCo
mmand[0]),
                L"{call %stpcc_stocklevel
(?,?,?)}", m_szSPPrefix);
}

void CTPCC_ODBC::StockLevel()
{
        RETCODE             rc;
        int                         iTryCount =
0;

        m_hstmt = m_hstmtStockLevel;

        while (TRUE)
        {
                try
                {
                        rc =
SQLExecDirectW(m_hstmt, m_szStockLevelCommand,
SQL_NTS);
                        if (rc != SQL_SUCCESS
&& rc != SQL_SUCCESS_WITH_INFO)

        ThrowError(CODBCERR::eExecDirect);

                        if ( SQLFetch(m_hstmt)
== SQL_ERROR )

        ThrowError(CODBCERR::eFetch);

                        SQLFreeStmt(m_hstmt,
SQL_CLOSE);
```

```
        m_txn.StockLevel.exec_status_code = eOK;
                        break;
                }
                catch (CODBCERR *e)
                {
                        if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                throw;

                        // hit deadlock;
backoff for increasingly longer period
                        delete e;
                        Sleep(10 * iTryCount);
                }
        }

//      if (iTryCount)
//              throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

void CTPCC_ODBC::InitNewOrderParams()
{
        if (   SQLAllocHandle(SQL_HANDLE_STMT,
m_hdbc, &m_hstmtNewOrder) != SQL_SUCCESS
                ||
SQLAllocHandle(SQL_HANDLE_STMT, m_hdbc,
&m_hstmtNewOrderNoDuplicates) != SQL_SUCCESS
                ||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descNewOrderCols1) != SQL_SUCCESS
                ||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descNewOrderCols2) != SQL_SUCCESS
                ||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descNewOrderNoDuplicatesCols1) != SQL_SUCCESS
                ||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descNewOrderNoDuplicatesCols2) != SQL_SUCCESS
                )

        ThrowError(CODBCERR::eAllocHandle);

        m_hstmt = m_hstmtNewOrder;

        if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC, m_descNewOrderCols1,
SQL_IS_POINTER ) != SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

        int i = 0;
        if (   SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.NewOrder.w_id, 0, NULL) != SQL_SUCCESS
                || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.NewOrder.d_id, 0, NULL) != SQL_SUCCESS
```

```
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.NewOrder.c_id, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.NewOrder.o_ol_cnt, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.NewOrder.o_all_local, 0, NULL) != SQL_SUCCESS
                    )
                    ThrowError(CODBCERR::eBindParam);

          for (int j=0; j<MAX_OL_NEW_ORDER_ITEMS;
j++)
          {
                    if (   SQLBindParameter(m_hstmt,
++i, SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.NewOrder.OL[j].ol_i_id, 0, NULL) !=
SQL_SUCCESS
                              ||
SQLBindParameter(m_hstmt, ++i, SQL_PARAM_INPUT,
SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.NewOrder.OL[j].ol_supply_w_id, 0, NULL) !=
SQL_SUCCESS
                              ||
SQLBindParameter(m_hstmt, ++i, SQL_PARAM_INPUT,
SQL_C_SSHORT, SQL_SMALLINT, 0, 0,
&m_txn.NewOrder.OL[j].ol_quantity, 0, NULL) !=
SQL_SUCCESS
                              )

          ThrowError(CODBCERR::eBindParam);
          }

          // set the bind offset pointer
          if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_ROW_BIND_OFFSET_PTR, &m_BindOffset,
SQL_IS_POINTER ) != SQL_SUCCESS )

          ThrowError(CODBCERR::eSetStmtAttr);

          i = 0;
          if (   SQLBindCol(m_hstmt, ++i, SQL_C_CHAR,
&m_txn.NewOrder.OL[0].ol_i_name,
sizeof(m_txn.NewOrder.OL[0].ol_i_name), NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_SSHORT, &m_txn.NewOrder.OL[0].ol_stock, 0,
NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,   &m_txn.NewOrder.OL[0].ol_brand_generic,
sizeof(m_txn.NewOrder.OL[0].ol_brand_generic), NULL)
!= SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE, &m_txn.NewOrder.OL[0].ol_i_price, 0,
NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE, &m_txn.NewOrder.OL[0].ol_amount, 0,
NULL) != SQL_SUCCESS
                    )
                    ThrowError(CODBCERR::eBindCol);
```

```
          // associate the column bindings for the
second result set
          if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC, m_descNewOrderCols2,
SQL_IS_POINTER ) != SQL_SUCCESS )

          ThrowError(CODBCERR::eSetStmtAttr);

          i = 0;
          if (   SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.NewOrder.w_tax, 0, NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.NewOrder.d_tax, 0, NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG,     &m_txn.NewOrder.o_id, 0, NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.NewOrder.c_last,
sizeof(m_txn.NewOrder.c_last), NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.NewOrder.c_discount, 0, NULL)
!= SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.NewOrder.c_credit,
sizeof(m_txn.NewOrder.c_credit), NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_TYPE_TIMESTAMP, &m_txn.NewOrder.o_entry_d, 0,
NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG,     &m_no_commit_flag, 0, NULL) !=
SQL_SUCCESS
               )
                    ThrowError(CODBCERR::eBindCol);

          //Compose the New Order statement
          _snwprintf(m_szNewOrderCommand,
sizeof(m_szNewOrderCommand)/sizeof(m_szNewOrderComman
d[0]),
                         // 0         1         2
                         //
01234567890123456789012345678 9
                              L"{call
%stpcc_neworder(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,"

          L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?)}", m_szSPPrefix);

          m_iBeginNewOrderVariablePart = 29 +
wcslen(m_szSPPrefix);         // fixed part + prefix
part

          ////////////////////////////////////////////
////////////////////////////////////////////
          //
          //        Now initialize New Order that
works on no duplicate (w_id,i_id) pairs
          //        and returns one result set for
lineitem details.
          //
          //
```

```
          m_hstmt = m_hstmtNewOrderNoDuplicates;

          if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descNewOrderNoDuplicatesCols1, SQL_IS_POINTER ) !=
SQL_SUCCESS )

          ThrowError(CODBCERR::eSetStmtAttr);

          i = 0;
          if (   SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.NewOrder.w_id, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.NewOrder.d_id, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.NewOrder.c_id, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.NewOrder.o_ol_cnt, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.NewOrder.o_all_local, 0, NULL) != SQL_SUCCESS
                    )
                    ThrowError(CODBCERR::eBindParam);

          for (int j=0; j<MAX_OL_NEW_ORDER_ITEMS;
j++)
          {
                    if (   SQLBindParameter(m_hstmt,
++i, SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.NewOrder.OL[j].ol_i_id, 0, NULL) !=
SQL_SUCCESS
                              ||
SQLBindParameter(m_hstmt, ++i, SQL_PARAM_INPUT,
SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.NewOrder.OL[j].ol_supply_w_id, 0, NULL) !=
SQL_SUCCESS
                              ||
SQLBindParameter(m_hstmt, ++i, SQL_PARAM_INPUT,
SQL_C_SSHORT, SQL_SMALLINT, 0, 0,
&m_txn.NewOrder.OL[j].ol_quantity, 0, NULL) !=
SQL_SUCCESS
                              )

          ThrowError(CODBCERR::eBindParam);
          }

          // set row-wise binding
          if (   SQLSetStmtAttrW(m_hstmt,
SQL_ATTR_ROW_BIND_TYPE,
(SQLPOINTER)sizeof(m_txn.NewOrder.OL[0]),
SQL_IS_UINTEGER) != SQL_SUCCESS
                    || SQLSetStmtAttrW(m_hstmt,
SQL_ATTR_ROWS_FETCHED_PTR, &m_RowsFetched, 0) !=
SQL_SUCCESS )

          ThrowError(CODBCERR::eSetStmtAttr);

          i = 0;
```

```
        if (   SQLBindCol(m_hstmt, ++i, SQL_C_CHAR,
&m_txn.NewOrder.OL[0].ol_i_name,
sizeof(m_txn.NewOrder.OL[0].ol_i_name), NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_SSHORT, &m_txn.NewOrder.OL[0].ol_stock, 0,
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,   &m_txn.NewOrder.OL[0].ol_brand_generic,
sizeof(m_txn.NewOrder.OL[0].ol_brand_generic), NULL)
!= SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE, &m_txn.NewOrder.OL[0].ol_i_price, 0,
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE, &m_txn.NewOrder.OL[0].ol_amount, 0,
NULL) != SQL_SUCCESS
                )
                ThrowError(CODBCERR::eBindCol);

        // associate the column bindings for the
second result set
        if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descNewOrderNoDuplicatesCols2, SQL_IS_POINTER ) !=
SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

        i = 0;
        if (   SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.NewOrder.w_tax, 0, NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.NewOrder.d_tax, 0, NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG,     &m_txn.NewOrder.o_id, 0, NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.NewOrder.c_last,
sizeof(m_txn.NewOrder.c_last), NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.NewOrder.c_discount, 0, NULL)
!= SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.NewOrder.c_credit,
sizeof(m_txn.NewOrder.c_credit), NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_TYPE_TIMESTAMP, &m_txn.NewOrder.o_entry_d, 0,
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG,     &m_no_commit_flag, 0, NULL) !=
SQL_SUCCESS
            )
                ThrowError(CODBCERR::eBindCol);

        //Compose the New Order statement
        _snwprintf(m_szNewOrderNoDuplicatesCommand,
sizeof(m_szNewOrderNoDuplicatesCommand)/sizeof(m_szNe
wOrderNoDuplicatesCommand[0]),
                        L"{call
%stpcc_neworder_new(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?,?,?,?,?,?,"

        L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?
,?,?,?,?)}", m_szSPPrefix);

        m_iBeginNewOrderNoDuplicatesVariablePart =
33 + wcslen(m_szSPPrefix);    // fixed part + prefix
part
}

//
//      Returns true if there are duplicate
(warehouse_id, item_id)
//      lineitem pairs in New Order input
parameters.
//
bool CTPCC_ODBC::DuplicatesInNewOrder()
{
        int i, j;

        for (i = 0; i < m_txn.NewOrder.o_ol_cnt;
++i)
        {
                for (j = i+1; j<
m_txn.NewOrder.o_ol_cnt; ++j)
                {
                        if
(m_txn.NewOrder.OL[i].ol_i_id ==
m_txn.NewOrder.OL[j].ol_i_id)
                        {
                                return true;
                        }
                }
        }

        return false;
}

void CTPCC_ODBC::NewOrder()
{
        if (m_bCallNoDuplicatesNewOrder)
        {
                if (DuplicatesInNewOrder())
                {
                        NewOrderDuplicates();
                }
                else
                {
                        NewOrderNoDuplicates();
                }
        }
        else
        {
                NewOrderDuplicates();
        }
}

void CTPCC_ODBC::NewOrderDuplicates()
{
        int
        i;
        RETCODE                                 rc;
        int
        iTryCount = 0;

                                                //
0       1       2

                                                //
01234567890123456789012345678
        wchar_t
        szSqlTemplate[iMAX_SP_NAME_LEN];

                                        //= L"{call
tpcc_neworder(?,?,?,?,?,"

                                        //
L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,"

                                        //
L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,"

                                        //
L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}";

        m_hstmt = m_hstmtNewOrder;

        // associate the parameter and column
bindings for this transaction
        if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC, m_descNewOrderCols1,
SQL_IS_POINTER ) != SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

        // clip statement buffer based on number of
parameters
        // fixed part is 29 chars and variable part
is 6 chars per line item
        wcscpy(szSqlTemplate, m_szNewOrderCommand);
        i = m_iBeginNewOrderVariablePart +
m_txn.NewOrder.o_ol_cnt*6;
        wcscpy( &szSqlTemplate[i], L")}" );

        // check whether any order lines are for a
remote warehouse
        m_txn.NewOrder.o_all_local = 1;
        for (i = 0; i < m_txn.NewOrder.o_ol_cnt;
i++)
        {
                if
(m_txn.NewOrder.OL[i].ol_supply_w_id !=
m_txn.NewOrder.w_id)
                {

        m_txn.NewOrder.o_all_local = 0;  // at
least one remote warehouse
                        break;
                }
        }

        while (TRUE)
        {
                try
```

```
                {
                        m_BindOffset = 0;
                        rc =
SQLExecDirectW(m_hstmt, szSqlTemplate, SQL_NTS);
                        if (rc != SQL_SUCCESS
&& rc != SQL_SUCCESS_WITH_INFO)

        ThrowError(CODBCERR::eExecDirect);

                        // Get order line
results

        m_txn.NewOrder.total_amount = 0;
                        for (i = 0;
i<m_txn.NewOrder.o_ol_cnt; i++)
                        {
                                // set the
bind offset value...
                                m_BindOffset
= i * sizeof(m_txn.NewOrder.OL[0]);

                                if (
SQLFetch(m_hstmt) == SQL_ERROR)

        ThrowError(CODBCERR::eFetch);

                                // move to
the next resultset
                                if (
SQLMoreResults(m_hstmt) == SQL_ERROR )

        ThrowError(CODBCERR::eMoreResults);


        m_txn.NewOrder.total_amount +=
m_txn.NewOrder.OL[i].ol_amount;
                        }

                        // associate the column
bindings for the second result set
                        if ( SQLSetStmtAttrW(
m_hstmt, SQL_ATTR_APP_ROW_DESC, m_descNewOrderCols2,
SQL_IS_POINTER ) != SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

                        if ( SQLFetch(m_hstmt)
== SQL_ERROR)

        ThrowError(CODBCERR::eFetch);

                        SQLFreeStmt(m_hstmt,
SQL_CLOSE);

                        if (m_no_commit_flag ==
1)
                        {

        m_txn.NewOrder.total_amount *= ((1 +
m_txn.NewOrder.w_tax + m_txn.NewOrder.d_tax) * (1 -
m_txn.NewOrder.c_discount));

        m_txn.NewOrder.exec_status_code = eOK;
                        }
                        else

        m_txn.NewOrder.exec_status_code =
eInvalidItem;

                        break;
                }
                catch (CODBCERR *e)
                {
                        if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                throw;

                        // hit deadlock;
backoff for increasingly longer period
                        delete e;
                        Sleep(10 * iTryCount);
                }
        }

//      if (iTryCount)
//              throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

//
//      No lineitem duplicates optimized version.
//
void CTPCC_ODBC::NewOrderNoDuplicates()
{
        int
        i;
        RETCODE                                 rc;
        int
        iTryCount = 0;


                                                //
0       1       2       3

                                                //
01234567890123456789012345678901234567890123
        wchar_t
        szSqlTemplate[iMAX_SP_NAME_LEN];

                                        //= L"{call
tpcc_neworder_new(?,?,?,?,?,"

                                        //
L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,"

                                        //
L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,"

                                        //
L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}";

        m_hstmt = m_hstmtNewOrderNoDuplicates;

        // associate the parameter and column
bindings for this transaction

        if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descNewOrderNoDuplicatesCols1, SQL_IS_POINTER ) !=
SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

        // clip statement buffer based on number of
parameters
        // fixed part is 33 chars and variable part
is 6 chars per line item
        wcscpy(szSqlTemplate,
m_szNewOrderNoDuplicatesCommand);
        i =
m_iBeginNewOrderNoDuplicatesVariablePart +
m_txn.NewOrder.o_ol_cnt*6;
        wcscpy( &szSqlTemplate[i], L")}" );

        // check whether any order lines are for a
remote warehouse
        m_txn.NewOrder.o_all_local = 1;
        for (i = 0; i < m_txn.NewOrder.o_ol_cnt;
i++)
        {
                if
(m_txn.NewOrder.OL[i].ol_supply_w_id !=
m_txn.NewOrder.w_id)
                {

        m_txn.NewOrder.o_all_local = 0;  // at
least one remote warehouse
                        break;
                }
        }

        while (TRUE)
        {
                try
                {
                        // configure block
cursor
                        if (
SQLSetStmtAttrW(m_hstmt, SQL_ATTR_ROW_ARRAY_SIZE,
(SQLPOINTER)1, 0) != SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

                        rc =
SQLExecDirectW(m_hstmt, szSqlTemplate, SQL_NTS);
                        if (rc != SQL_SUCCESS
&& rc != SQL_SUCCESS_WITH_INFO)

        ThrowError(CODBCERR::eExecDirect);

                        // configure block
cursor
                        if
(SQLSetStmtAttrW(m_hstmt, SQL_ATTR_ROW_ARRAY_SIZE,
(SQLPOINTER)MAX_OL_NEW_ORDER_ITEMS, 0) !=
SQL_SUCCESS)

        ThrowError(CODBCERR::eSetStmtAttr);
```

```
                            // Get order line
results
                            if ( SQLFetch(m_hstmt)
== SQL_ERROR)
          ThrowError(CODBCERR::eFetch);



          m_txn.NewOrder.total_amount = 0;
                            for (i = 0;
i<m_txn.NewOrder.o_ol_cnt; i++)
                            {


          m_txn.NewOrder.total_amount +=
m_txn.NewOrder.OL[i].ol_amount;
                            }

                            // associate the column
bindings for the second result set
                            if ( SQLSetStmtAttrW(
m_hstmt, SQL_ATTR_APP_ROW_DESC,
m_descNewOrderNoDuplicatesCols2, SQL_IS_POINTER ) !=
SQL_SUCCESS )
          ThrowError(CODBCERR::eSetStmtAttr);

                            // move to the next
resultset
                            if (
SQLMoreResults(m_hstmt) == SQL_ERROR )
          ThrowError(CODBCERR::eMoreResults);

                            if ( (rc =
SQLFetch(m_hstmt)) == SQL_ERROR)
          ThrowError(CODBCERR::eFetch);

                            SQLFreeStmt(m_hstmt,
SQL_CLOSE);

                            // Check Fetch return
code for no rows returned.
                            // It means customer id
or warehouse id were invalid.
                            //
                            if (rc == SQL_NO_DATA)
                                  throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_INVALID_NEW_ORDER_
PARAM);

                            if (m_no_commit_flag ==
1)
                            {

          m_txn.NewOrder.total_amount *= ((1 +
m_txn.NewOrder.w_tax + m_txn.NewOrder.d_tax) * (1 -
m_txn.NewOrder.c_discount));

          m_txn.NewOrder.exec_status_code = eOK;
                            }
                            else

          m_txn.NewOrder.exec_status_code =
eInvalidItem;

                            break;
                      }
                      catch (CODBCERR *e)
                      {
                            if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                  throw;

                            // hit deadlock;
backoff for increasingly longer period
                            delete e;
                            Sleep(10 * iTryCount);
                      }
          }

//        if (iTryCount)
//              throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

void CTPCC_ODBC::InitPaymentParams()
{
          if (   SQLAllocHandle(SQL_HANDLE_STMT,
m_hdbc, &m_hstmtPayment) != SQL_SUCCESS )

          ThrowError(CODBCERR::eAllocHandle);

          m_hstmt = m_hstmtPayment;

          int i = 0;
          if (   SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.Payment.w_id, 0, NULL) != SQL_SUCCESS
                      || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.Payment.c_w_id, 0, NULL) != SQL_SUCCESS
                      || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_DOUBLE, SQL_NUMERIC, 6, 2,
&m_txn.Payment.h_amount, 0, NULL) != SQL_SUCCESS
                      || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.Payment.d_id, 0, NULL) != SQL_SUCCESS
                      || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.Payment.c_d_id, 0, NULL) != SQL_SUCCESS
                      || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.Payment.c_id, 0, NULL) != SQL_SUCCESS
                      || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR,
sizeof(m_txn.Payment.c_last), 0,
&m_txn.Payment.c_last, sizeof(m_txn.Payment.c_last),
NULL) != SQL_SUCCESS
                      )
                      ThrowError(CODBCERR::eBindParam);

          i = 0;
          if (   SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG,     &m_txn.Payment.c_id,             0,
NULL) != SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_last,
          sizeof(m_txn.Payment.c_last), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_TYPE_TIMESTAMP, &m_txn.Payment.h_date,
          0, NULL) != SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.w_street_1,
          sizeof(m_txn.Payment.w_street_1), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.w_street_2,
          sizeof(m_txn.Payment.w_street_2), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.w_city,
          sizeof(m_txn.Payment.w_city), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.w_state,
          sizeof(m_txn.Payment.w_state), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.w_zip,
          sizeof(m_txn.Payment.w_zip), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.d_street_1,
          sizeof(m_txn.Payment.d_street_1), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.d_street_2,
          sizeof(m_txn.Payment.d_street_2), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.d_city,
          sizeof(m_txn.Payment.d_city), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.d_state,
          sizeof(m_txn.Payment.d_state), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.d_zip,
          sizeof(m_txn.Payment.d_zip), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_first,
          sizeof(m_txn.Payment.c_first), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_middle,
          sizeof(m_txn.Payment.c_middle), NULL) !=
SQL_SUCCESS
                      || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_street_1,
          sizeof(m_txn.Payment.c_street_1), NULL) !=
SQL_SUCCESS
```

```
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_street_2,
          sizeof(m_txn.Payment.c_street_2), NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_city,
          sizeof(m_txn.Payment.c_city), NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_state,
          sizeof(m_txn.Payment.c_state), NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_zip,
          sizeof(m_txn.Payment.c_zip), NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_phone,
          sizeof(m_txn.Payment.c_phone), NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_TYPE_TIMESTAMP, &m_txn.Payment.c_since,
          0, NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_credit,
          sizeof(m_txn.Payment.c_credit), NULL) !=
SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.Payment.c_credit_lim,0, NULL)
!= SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.Payment.c_discount,       0,
NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.Payment.c_balance,        0,
NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,      &m_txn.Payment.c_data,
          sizeof(m_txn.Payment.c_data), NULL) !=
SQL_SUCCESS
                    )
                    ThrowError(CODBCERR::eBindCol);

          //Compose Payment statement
          _snwprintf(m_szPaymentCommand,
sizeof(m_szPaymentCommand)/sizeof(m_szPaymentCommand[
0]),
                    L"{call %stpcc_payment
(?,?,?,?,?,?,?,?)}", m_szSPPrefix);
}

void CTPCC_ODBC::Payment()
{
          RETCODE             rc;
          int                           iTryCount =
0;

          m_hstmt = m_hstmtPayment;

          if (m_txn.Payment.c_id != 0)
                    m_txn.Payment.c_last[0] = 0;

          while (TRUE)
          {
                    try
                    {
                              rc =
SQLExecDirectW(m_hstmt, m_szPaymentCommand, SQL_NTS);
                              if (rc != SQL_SUCCESS
&& rc != SQL_SUCCESS_WITH_INFO)

          ThrowError(CODBCERR::eExecDirect);

                              if ( SQLFetch(m_hstmt)
== SQL_ERROR)

          ThrowError(CODBCERR::eFetch);

                              SQLFreeStmt(m_hstmt,
SQL_CLOSE);

                              if (m_txn.Payment.c_id
== 0)
                                        throw new
CTPCC_ODBC_ERR( CTPCC_ODBC_ERR::ERR_INVALID_CUST );
                              else

          m_txn.Payment.exec_status_code = eOK;

                              break;
                    }
                    catch (CODBCERR *e)
                    {
                              if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                        throw;

                              // hit deadlock;
backoff for increasingly longer period
                              delete e;
                              Sleep(10 * iTryCount);
                    }
          }

//        if (iTryCount)
//                  throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

void CTPCC_ODBC::InitOrderStatusParams()
{
          if (   SQLAllocHandle(SQL_HANDLE_STMT,
m_hdbc, &m_hstmtOrderStatus) != SQL_SUCCESS
                    ||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descOrderStatusCols1) != SQL_SUCCESS
                    ||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descOrderStatusCols2) != SQL_SUCCESS
                    )

          ThrowError(CODBCERR::eAllocHandle);

          m_hstmt = m_hstmtOrderStatus;

          if (   SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC, m_descOrderStatusCols1,
SQL_IS_POINTER ) != SQL_SUCCESS )

          ThrowError(CODBCERR::eSetStmtAttr);

          int i = 0;
          if (   SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.OrderStatus.w_id, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT, SQL_TINYINT, 0, 0,
&m_txn.OrderStatus.d_id, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.OrderStatus.c_id, 0, NULL) != SQL_SUCCESS
                    || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR,
sizeof(m_txn.OrderStatus.c_last), 0,
&m_txn.OrderStatus.c_last,
sizeof(m_txn.OrderStatus.c_last), NULL) !=
SQL_SUCCESS
                    )
                    ThrowError(CODBCERR::eBindParam);

          // configure block cursor
          if (   SQLSetStmtAttrW(m_hstmt,
SQL_ATTR_ROW_BIND_TYPE,
(SQLPOINTER)sizeof(m_txn.OrderStatus.OL[0]), 0) !=
SQL_SUCCESS
                    || SQLSetStmtAttrW(m_hstmt,
SQL_ATTR_ROWS_FETCHED_PTR, &m_RowsFetched, 0) !=
SQL_SUCCESS
                    )

          ThrowError(CODBCERR::eSetStmtAttr);

          i = 0;
          if (   SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG, &m_txn.OrderStatus.OL[0].ol_supply_w_id,
0, NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG, &m_txn.OrderStatus.OL[0].ol_i_id, 0,
NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_SSHORT, &m_txn.OrderStatus.OL[0].ol_quantity,
0, NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE, &m_txn.OrderStatus.OL[0].ol_amount, 0,
NULL) != SQL_SUCCESS
                    || SQLBindCol(m_hstmt, ++i,
SQL_C_TYPE_TIMESTAMP,
&m_txn.OrderStatus.OL[0].ol_delivery_d, 0, NULL) !=
SQL_SUCCESS
                    )
                    ThrowError(CODBCERR::eBindCol);


          if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC, m_descOrderStatusCols2,
SQL_IS_POINTER ) != SQL_SUCCESS )

          ThrowError(CODBCERR::eSetStmtAttr);
```

```
        i = 0;
        if (   SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG, &m_txn.OrderStatus.c_id, 0, NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR, &m_txn.OrderStatus.c_last,
sizeof(m_txn.OrderStatus.c_last), NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR, &m_txn.OrderStatus.c_first,
sizeof(m_txn.OrderStatus.c_first), NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR, &m_txn.OrderStatus.c_middle,
sizeof(m_txn.OrderStatus.c_middle), NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_TYPE_TIMESTAMP, &m_txn.OrderStatus.o_entry_d,
0, NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_SSHORT, &m_txn.OrderStatus.o_carrier_id, 0,
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE, &m_txn.OrderStatus.c_balance, 0, NULL)
!= SQL_SUCCESS
                || SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG, &m_txn.OrderStatus.o_id, 0, NULL) !=
SQL_SUCCESS
                )
                ThrowError(CODBCERR::eBindCol);

        //Compose Order Status statement
        _snwprintf(m_szOrderStatusCommand,
sizeof(m_szOrderStatusCommand)/sizeof(m_szOrderStatus
Command[0]),
                L"{call %stpcc_orderstatus
(?,?,?,?)}", m_szSPPrefix);
}

void CTPCC_ODBC::OrderStatus()
{
        int
                iTryCount = 0;
        RETCODE
        rc;

        m_hstmt = m_hstmtOrderStatus;

        if (m_txn.OrderStatus.c_id != 0)
                m_txn.OrderStatus.c_last[0] = 0;

        while (TRUE)
        {
                try
                {
                        if ( SQLSetStmtAttrW(
m_hstmt, SQL_ATTR_APP_ROW_DESC,
m_descOrderStatusCols1, SQL_IS_POINTER ) !=
SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);
```

```
                        // configure block
cursor
                        if (
SQLSetStmtAttrW(m_hstmt, SQL_ATTR_ROW_ARRAY_SIZE,
(SQLPOINTER)1, 0) != SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

                        rc =
SQLExecDirectW(m_hstmt, m_szOrderStatusCommand,
SQL_NTS);
                        if (rc != SQL_SUCCESS
&& rc != SQL_SUCCESS_WITH_INFO)

        ThrowError(CODBCERR::eExecDirect);


                        // configure block
cursor
                        if (
SQLSetStmtAttrW(m_hstmt, SQL_ATTR_ROW_ARRAY_SIZE,
(SQLPOINTER)MAX_OL_ORDER_STATUS_ITEMS, 0) !=
SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

                        rc = SQLFetchScroll(
m_hstmt, SQL_FETCH_NEXT, 0 );
//                      if ( !((rc ==
SQL_SUCCESS) || ((rc == SQL_SUCCESS_WITH_INFO) &&
(m_RowsFetched != 0))) )
                        if ( (rc !=
SQL_SUCCESS) )

        ThrowError(CODBCERR::eFetchScroll);


        m_txn.OrderStatus.o_ol_cnt =
(short)m_RowsFetched;

                        if
(m_txn.OrderStatus.o_ol_cnt != 0)
                        {
                                if (
SQLSetStmtAttrW( m_hstmt, SQL_ATTR_APP_ROW_DESC,
m_descOrderStatusCols2, SQL_IS_POINTER ) !=
SQL_SUCCESS )

        ThrowError(CODBCERR::eSetStmtAttr);

//                              if (
SQLMoreResults(m_hstmt) == SQL_ERROR )
                                if ( (rc =
SQLMoreResults(m_hstmt)) != SQL_SUCCESS )
                                {

        ThrowError(CODBCERR::eMoreResults);
                                }

//                              if ( (rc =
SQLFetch(m_hstmt)) == SQL_ERROR)
```

```
                                if ( (rc =
SQLFetch(m_hstmt)) != SQL_SUCCESS)

        ThrowError(CODBCERR::eFetch);
                                }

                                SQLFreeStmt(m_hstmt,
SQL_CLOSE);

                                if
(m_txn.OrderStatus.o_ol_cnt == 0)
                                        throw new
CTPCC_ODBC_ERR( CTPCC_ODBC_ERR::ERR_NO_SUCH_ORDER );
                                else if
(m_txn.OrderStatus.c_id == 0 &&
m_txn.OrderStatus.c_last[0] == 0)
                                        throw new
CTPCC_ODBC_ERR( CTPCC_ODBC_ERR::ERR_INVALID_CUST );
                                else

        m_txn.OrderStatus.exec_status_code = eOK;

                                break;
                        }
                        catch (CODBCERR *e)
                        {
                                if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                        throw;

                                // hit deadlock;
backoff for increasingly longer period
                                delete e;
                                Sleep(10 * iTryCount);
                        }
                }

//              if (iTryCount)
//                      throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

void CTPCC_ODBC::InitDeliveryParams()
{
        if (   SQLAllocHandle(SQL_HANDLE_STMT,
m_hdbc, &m_hstmtDelivery) != SQL_SUCCESS )

        ThrowError(CODBCERR::eAllocHandle);

        m_hstmt = m_hstmtDelivery;

        int i = 0;
        if (   SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG, SQL_INTEGER, 0, 0,
&m_txn.Delivery.w_id, 0, NULL) != SQL_SUCCESS
                || SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SSHORT, SQL_SMALLINT, 0, 0,
&m_txn.Delivery.o_carrier_id, 0, NULL) != SQL_SUCCESS
                )
                ThrowError(CODBCERR::eBindParam);

        for (i=0;i<10;i++)
```

```
        {
                if ( SQLBindCol(m_hstmt,
(UWORD)(i+1), SQL_C_SLONG, &m_txn.Delivery.o_id[i],
0, NULL) != SQL_SUCCESS )

        ThrowError(CODBCERR::eBindCol);
        }

        //Compose Delivery statement
        _snwprintf(m_szDeliveryCommand,
sizeof(m_szDeliveryCommand)/sizeof(m_szDeliveryComman
d[0]),
                L"{call %stpcc_delivery (?,?)}",
m_szSPPrefix);
}

void CTPCC_ODBC::Delivery()
{
        RETCODE             rc;
        int                         iTryCount =
0;

        m_hstmt = m_hstmtDelivery;

        while (TRUE)
        {
                try
                {
                        rc =
SQLExecDirectW(m_hstmt, m_szDeliveryCommand,
SQL_NTS);
                        if (rc != SQL_SUCCESS
&& rc != SQL_SUCCESS_WITH_INFO)

        ThrowError(CODBCERR::eExecDirect);

                        if ( SQLFetch(m_hstmt)
== SQL_ERROR )

        ThrowError(CODBCERR::eFetch);

                        SQLFreeStmt(m_hstmt,
SQL_CLOSE);

        m_txn.Delivery.exec_status_code = eOK;
                        break;
                }
                catch (CODBCERR *e)
                {
                        if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                throw;

                        // hit deadlock;
backoff for increasingly longer period
                        delete e;
                        Sleep(10 * iTryCount);
                }
        }

//      if (iTryCount)
//                throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIED_TRANS,
iTryCount);
}
```

tpcc_odbc.h

```
/*      FILE:               TPCC_ODBC.H
 *                                      Microsoft
TPC-C Kit Ver. 4.69.000
 *                                      Copyright
Microsoft, 1999
 *             All Rights Reserved
 *
 *                                      Version
4.10.000 audited by Richard Gimarc, Performance
Metrics, 3/17/99
 *
 *      PURPOSE:  Header file for TPC-C txn class
implementation.
 *
 *  Change history:
 *                  4.20.000 - updated rev number to
match kit
 *                  4.69.000 - updated rev number to
match kit
 */
#pragma once

// need to declare functions for import, unless
define has already been created
// by the DLL's .cpp module for export.
#ifndef DllDecl
#define DllDecl  __declspec( dllimport )
#endif

#define iMAX_SP_NAME_LEN 256  //maximum length of a
stored procedure name with parameters

class CODBCERR : public CBaseErr
{
        public:
                enum ACTION
                {
                        eNone,
                        eUnknown,
                        eAllocConn,
                // error from SQLAllocConnect
                        eAllocHandle,
        // error from SQLAllocHandle
                        eConnOption,
        // error from SQLSetConnectOption
                        eConnect,
        // error from SQLConnect
                        eAllocStmt,
                // error from SQLAllocStmt
                        eExecDirect,
        // error from SQLExecDirect
                        eBindParam,
                // error from SQLBindParameter
                        eBindCol,
        // error from SQLBindCol
                        eFetch,
                 // error from SQLFetch
                        eFetchScroll,
        // error from SQLFetchScroll
                        eMoreResults,
        // error from SQLMoreResults
                        ePrepare,
        // error from SQLPrepare
                        eExecute,
        // error from SQLExecute
                        eSetEnvAttr,
        // error from SQLSetEnvAttr
                        eSetStmtAttr
        // error from SQLSetStmtAttr
                };

                CODBCERR(void)
                {
                        m_eAction = eNone;
                        m_NativeError = 0;
                        m_bDeadLock = FALSE;
                        m_odbcerrstr = NULL;
                };

                ~CODBCERR()
                {
                        if (m_odbcerrstr !=
NULL)
                                delete []
m_odbcerrstr;
                };

                ACTION    m_eAction;
                int
        m_NativeError;
                BOOL      m_bDeadLock;
                char   *m_odbcerrstr;

                int                 ErrorType()
{return ERR_TYPE_ODBC;};
                char*     ErrorTypeStr() { return
"ODBC"; }
                int                 ErrorNum()
{return m_NativeError;};
                char*     ErrorText() {return
m_odbcerrstr;};
                int                 ErrorAction()
{ return (int)m_eAction; }
};

class CTPCC_ODBC_ERR : public CBaseErr
{
        public:
                enum TPCC_ODBC_ERRS
                {
                        ERR_WRONG_SP_VERSION =
1,      // "Wrong version of stored procs on
database server"
                        ERR_INVALID_CUST,
                // "Invalid Customer id,name."
```

```
                        ERR_NO_SUCH_ORDER,
                // "No orders found for
customer."
                        ERR_RETRIED_TRANS,
                // "Retries before transaction
succeeded."

        ERR_INVALID_NEW_ORDER_PARAM   // "New Order
parameter invalid."
                };

                CTPCC_ODBC_ERR( int iErr ) {
m_errno = iErr; m_iTryCount = 0; };

                CTPCC_ODBC_ERR( int iErr, int
iTryCount ) { m_errno = iErr; m_iTryCount =
iTryCount; };

                int             m_errno;
                int             m_iTryCount;

                int             ErrorType()
{return ERR_TYPE_TPCC_ODBC;};
                char*   ErrorTypeStr() { return
"TPCC ODBC"; }
                int             ErrorNum()
{return m_errno;};

                char*   ErrorText();
};

class DllDecl CTPCC_ODBC : public CTPCC_BASE
{
        private:
                // declare variables and private
functions here...
                BOOL            m_bDeadlock;
                // transaction was selected as
deadlock victim
                int
        m_MaxRetries;           // retry
count on deadlock

                SQLHENV         m_henv;
                        // ODBC environment
handle
                SQLHDBC         m_hdbc;
                SQLHSTMT  m_hstmt;
        // the current hstmt

                SQLHSTMT  m_hstmtNewOrder;
                SQLHSTMT
        m_hstmtNewOrderNoDuplicates;  // NewOrder
with one result set for lineitem details
                SQLHSTMT  m_hstmtPayment;
                SQLHSTMT  m_hstmtDelivery;
                SQLHSTMT  m_hstmtOrderStatus;
                SQLHSTMT  m_hstmtStockLevel;

                SQLHDESC  m_descNewOrderCols1;
                SQLHDESC  m_descNewOrderCols2;
                SQLHDESC
        m_descNewOrderNoDuplicatesCols1;        //
NewOrder with one result set for lineitem details
                SQLHDESC
        m_descNewOrderNoDuplicatesCols2;        //
NewOrder with one result set for lineitem details
                SQLHDESC  m_descOrderStatusCols1;
                SQLHDESC  m_descOrderStatusCols2;

                wchar_t
        m_szSPPrefix[32];   // stored procedures
prefix

                wchar_t
        m_szNewOrderCommand[iMAX_SP_NAME_LEN];
                wchar_t
        m_szNewOrderNoDuplicatesCommand[iMAX_SP_NAM
E_LEN];
                int
        m_iBeginNewOrderVariablePart; // begining
of the variable part in NewOrder statement
                int
        m_iBeginNewOrderNoDuplicatesVariablePart;
        // begining of the variable part in
NewOrder statement
                wchar_t
        m_szPaymentCommand[iMAX_SP_NAME_LEN];
                wchar_t
        m_szDeliveryCommand[iMAX_SP_NAME_LEN];
                wchar_t
        m_szOrderStatusCommand[iMAX_SP_NAME_LEN];
                wchar_t
        m_szStockLevelCommand[iMAX_SP_NAME_LEN];

                // new-order specific fields
                SQLUINTEGER         m_BindOffset;
                SQLUINTEGER
        m_RowsFetched;
                int
        m_no_commit_flag;

                // tpcc_neworder_new flag
                BOOL
        m_bCallNoDuplicatesNewOrder;

                //void ThrowError(
CODBCERR::ACTION eAction );
                void ThrowError( RETCODE eAction
);

                void InitNewOrderParams();
                void InitPaymentParams();
                void InitDeliveryParams();
                void InitStockLevelParams();
                void InitOrderStatusParams();

                union
                {
                        NEW_ORDER_DATA
        NewOrder;
                        PAYMENT_DATA
        Payment;
                        DELIVERY_DATA
        Delivery;
                        STOCK_LEVEL_DATA
        StockLevel;
                        ORDER_STATUS_DATA
        OrderStatus;
                }
        m_txn;

                bool DuplicatesInNewOrder();
                void NewOrderDuplicates();
                void NewOrderNoDuplicates();

        public:
                CTPCC_ODBC(         LPCSTR
szServer, LPCSTR szUser, LPCSTR szPassword,

        LPCSTR szHost, LPCSTR szDatabase,

        LPCWSTR szSPPrefix, BOOL
bCallNoDuplicatesNewOrder);
                ~CTPCC_ODBC(void);

                inline PNEW_ORDER_DATA
        BuffAddr_NewOrder()           { return
&m_txn.NewOrder;    };
                inline PPAYMENT_DATA
        BuffAddr_Payment()            { return
&m_txn.Payment;     };
                inline PDELIVERY_DATA
        BuffAddr_Delivery()           { return
&m_txn.Delivery;    };
                inline PSTOCK_LEVEL_DATA
        BuffAddr_StockLevel()         { return
&m_txn.StockLevel;  };
                inline PORDER_STATUS_DATA
        BuffAddr_OrderStatus()        { return
&m_txn.OrderStatus; };

                void NewOrder                 ();
                void Payment                  ();
                void Delivery                 ();
                void StockLevel               ();
                void OrderStatus    ();

};


// wrapper routine for class constructor
extern "C" DllDecl CTPCC_ODBC* CTPCC_ODBC_new
          (         LPCSTR szServer, LPCSTR szUser,
LPCSTR szPassword,
                LPCSTR szHost, LPCSTR szDatabase,
                LPCWSTR szSPPrefix, BOOL
bCallNoDuplicatesNewOrder );

typedef CTPCC_ODBC* (TYPE_CTPCC_ODBC)(LPCSTR, LPCSTR,
LPCSTR, LPCSTR, LPCSTR, LPCWSTR, BOOL);
```

tpcc_oledb.cpp

```
/*        FILE:               TPCC_OLEDB.CPP
 *                                    Microsoft
TPC-C Kit Ver. 4.69.000
 *                                    Copyright
Microsoft, 2004
 *                                    Written by
Sergey Vasilevskiy
 *              All Rights Reserved
 *
 *
 *
 *        PURPOSE:  Implements OLEDB calls for TPC-C
txns.
 *        Contact:  Charles Levine
(clevine@microsoft.com)
 *
 *
 *                 4.69.000 - updated rev number to
match kit
 */

#include <windows.h>
#include <stdio.h>
#include <assert.h>
#include <stddef.h>

#define DBINITCONSTANTS
#include <oledb.h>
//#include <sqloledb.h> // Use MDAC
#include <C:\\Program Files\\Microsoft SQL
Server\\100\\SDK\\Include\\sqlncli.h>   // Use SNAC
#include <oledberr.h>

#ifdef ICECAP
#include <icapexp.h>
#endif

// need to declare functions for export
#define DllDecl  __declspec( dllexport )

#include "..\..\common\src\error.h"
#include "..\..\common\src\trans.h"
#include "..\..\common\src\txn_base.h"
#include "tpcc_oledb.h"

#ifndef SQL_MAX_MESSAGE_LENGTH
#define SQL_MAX_MESSAGE_LENGTH 512
#endif

// version string; must match return value from
tpcc_version stored proc
const char          sVersion[] = "4.20.000";

const iMaxRetries = 10;                 // how many
retries on deadlock

const int iErrOleDbProvider = 7312;
const char sErrTimeoutExpired[] = "Timeout expired";

// this needs to be the same as the max length of
machine/database/user/password in Benchcraft
(engstut.h)
const static int    iMaxNameLen = 32;

BOOL APIENTRY DllMain(HMODULE hModule, DWORD
ul_reason_for_call, LPVOID lpReserved)
{
          switch( ul_reason_for_call )
          {
                    case DLL_PROCESS_ATTACH:

          DisableThreadLibraryCalls(hModule);

                              break;

                    case DLL_PROCESS_DETACH:

                              break;

                    default:
                              /* nothing */;
          }
          return TRUE;
}

/* FUNCTION: CTPCC_OLEDB_ERR::ErrorText
 *
 */

char* CTPCC_OLEDB_ERR::ErrorText(void)
{
          int i;

          static SERRORMSG errorMsgs[] =
          {
                    { ERR_WRONG_SP_VERSION,
          "Wrong version of stored procs on database
server"   },
                    { ERR_INVALID_CUST,
          "Invalid Customer id,name."
                                                       },
                    { ERR_NO_SUCH_ORDER,
          "No orders found for customer."
                                                       },
                    { ERR_RETRIED_TRANS,
          "Retries before transaction succeeded."
                              },
                    { 0,
                              ""



          }
          };

          static char szNotFound[] = "Unknown error
number.";

          for(i=0; errorMsgs[i].szMsg[0]; i++)
          {
                    if ( m_errno ==
errorMsgs[i].iError )
                              break;
          }
          if ( !errorMsgs[i].szMsg[0] )
                    return szNotFound;
          else
                    return errorMsgs[i].szMsg;
}

// wrapper routine for class constructor
__declspec(dllexport) CTPCC_OLEDB* CTPCC_OLEDB_new(
          LPCSTR szServer,              // name of
SQL server
          LPCSTR szUser,                              //
user name for login
          LPCSTR szPassword,            // password
for login
          LPCSTR szHost,                              //
not used
          LPCSTR szDatabase,            // name of
database to use
          LPCWSTR szSPPrefix )                        //
prefix to append to the stored procedure names
{
          return new CTPCC_OLEDB( szServer, szUser,
szPassword, szHost, szDatabase, szSPPrefix );
}


CTPCC_OLEDB::CTPCC_OLEDB (
          LPCSTR szServer,
          // name of SQL server
          LPCSTR szUser,
                    // user name for login
          LPCSTR szPassword,
          // password for login
          LPCSTR szHost,
                    // not used
          LPCSTR szDatabase,
          // name of database to use
          LPCWSTR szSPPrefix
          // prefix to append to the stored procedure
names
          )
          : m_pIMalloc(NULL)
{
          int
          iRc;
          int
          i;
          HRESULT                                     hr;

          IDBInitialize*
          pIDBInitialize = NULL;                      //
data source interface
          IDBProperties*
          pIDBProperties = NULL;
          ICommandText*
          pICommandText;
          // SQL command without parameters
          wchar_t
          szwServer[iMaxNameLen];                     //
Unicode string used to convert to BSTR
```

```
          wchar_t
          szDatabase[iMaxNameLen];     // Unicode
string used to convert to BSTR
          wchar_t
          szUser[iMaxNameLen];                    //
Unicode string used to convert to BSTR
          wchar_t
          szPassword[iMaxNameLen];     // Unicode
string used to convert to BSTR

          // Copy stored procedures prefix
          wcsncpy(m_szSPPrefix, szSPPrefix,
sizeof(m_szSPPrefix)/sizeof(m_szSPPrefix[0]));

          // Convert single byte ANSI strings to
Unicode (for later conversion to BSTR)
          iRc = MultiByteToWideChar(CP_THREAD_ACP,
MB_PRECOMPOSED, szServer, (int)strlen(szServer)+1,
szwServer, iMaxNameLen);
          iRc = MultiByteToWideChar(CP_THREAD_ACP,
MB_PRECOMPOSED, szDatabase,
(int)strlen(szDatabase)+1, szwDatabase, iMaxNameLen);
          iRc = MultiByteToWideChar(CP_THREAD_ACP,
MB_PRECOMPOSED, szUser, (int)strlen(szUser)+1,
szwUser, iMaxNameLen);
          iRc = MultiByteToWideChar(CP_THREAD_ACP,
MB_PRECOMPOSED, szPassword,
(int)strlen(szPassword)+1, szwPassword, iMaxNameLen);

          // Initialize COM library to be able to use
OLE-DB interfaces
          CoInitialize(NULL);

          // Initialization - create SQLOLEDB
component
          //hr = CoCreateInstance(CLSID_SQLOLEDB, //
GUID of SQLOLEDB component
          // Compile for SNAC
          hr = CoCreateInstance(CLSID_SQLNCLI,    //
GUID of SQLNCLI component
                              NULL,
                        // not defining an aggregate
component, so NULL
                              CLSCTX_INPROC_SERVER,   //
run the component in our process
                              IID_IDBInitialize,
                              (void **) &pIDBInitialize);
    /*
    Initialize the property values needed
    to establish the connection.
    */
    for(i = 0; i < 4; i++)
        VariantInit(&m_InitProperties[i].vValue);
    //Server name.
    m_InitProperties[0].dwPropertyID  =
DBPROP_INIT_DATASOURCE;
    m_InitProperties[0].vValue.vt     = VT_BSTR;
    m_InitProperties[0].vValue.bstrVal=
SysAllocString(szwServer);
    m_InitProperties[0].dwOptions     =
DBPROPOPTIONS_REQUIRED;
    m_InitProperties[0].colid         = DB_NULLID;
    //Database.
    m_InitProperties[1].dwPropertyID  =
DBPROP_INIT_CATALOG;
    m_InitProperties[1].vValue.vt     = VT_BSTR;
    m_InitProperties[1].vValue.bstrVal=
SysAllocString(szwDatabase);
    m_InitProperties[1].dwOptions     =
DBPROPOPTIONS_REQUIRED;
    m_InitProperties[1].colid         = DB_NULLID;
    //Username (login).
    m_InitProperties[2].dwPropertyID  =
DBPROP_AUTH_USERID;
    m_InitProperties[2].vValue.vt     = VT_BSTR;
    m_InitProperties[2].vValue.bstrVal=
SysAllocString(szwUser);
    m_InitProperties[2].dwOptions     =
DBPROPOPTIONS_REQUIRED;
    m_InitProperties[2].colid         = DB_NULLID;
    //Password.
    m_InitProperties[3].dwPropertyID  =
DBPROP_AUTH_PASSWORD;
    m_InitProperties[3].vValue.vt     = VT_BSTR;
    m_InitProperties[3].vValue.bstrVal=
SysAllocString(szwPassword);
    m_InitProperties[3].dwOptions     =
DBPROPOPTIONS_REQUIRED;
    m_InitProperties[3].colid         = DB_NULLID;
    /*
    Construct the DBPROPSET
structure(m_rgInitPropSet). The
    DBPROPSET structure is used to pass an array of
DBPROP
    structures (m_InitProperties) to the
SetProperties method.
    */
    m_rgInitPropSet.guidPropertySet =
DBPROPSET_DBINIT;
    m_rgInitPropSet.cProperties    = 4;
    m_rgInitPropSet.rgProperties   =
m_InitProperties;
    //Set initialization properties.
          if (FAILED(hr = pIDBInitialize-
>QueryInterface(IID_IDBProperties,


                        (void **)&pIDBProperties)))
          {
                        ThrowError(pIDBInitialize,
COLEDBERR::eQueryInterface, "CTPCC_OLEDB()");
          }

    hr = pIDBProperties->SetProperties(1,
&m_rgInitPropSet);

    pIDBProperties->Release();
    //Now establish the connection to the data
source.
    hr = pIDBInitialize->Initialize();

          // Free BSTR property strings
          for(i = 0; i < 4; i++)
          {
SysFreeString(m_InitProperties[i].vValue.bstrVal);

          }

          hr = pIDBInitialize-
>QueryInterface(IID_IDBCreateSession, (void
**)&m_pIDBCreateSession);

          // Releasing this has no effect on the SQL
Server connection
     // of the data source object because of the
reference maintained by
     // m_pIDBCreateSession.
     pIDBInitialize->Release();
     pIDBInitialize = NULL;

          hr = m_pIDBCreateSession-
>CreateSession(NULL, IID_IDBCreateCommand, (IUnknown
**)&m_pIDBCreateCommand);
          if (FAILED(hr))
          {
                        ThrowError(m_pIDBCreateSession,
COLEDBERR::eCreateSession, "CTPCC_OLEDB()");
          }

          hr = m_pIDBCreateCommand-
>CreateCommand(NULL, IID_ICommandText, (IUnknown
**)&pICommandText);
          if (FAILED(hr))
          {
                        ThrowError(m_pIDBCreateCommand,
COLEDBERR::eCreateCommand, "CTPCC_OLEDB()");
          }

          hr = pICommandText-
>SetCommandText(DBGUID_SQL, L"set nocount on set
XACT_ABORT ON");
          if (FAILED(hr))
          {
                        ThrowError(pICommandText,
COLEDBERR::eSetCommandText, "CTPCC_OLEDB()");
          }

          hr = pICommandText->Execute(NULL, IID_NULL,
NULL, NULL, NULL);
          if (FAILED(hr))
          {
                        ThrowError(pICommandText,
COLEDBERR::eExecute, "CTPCC_OLEDB()");
          }

          pICommandText->Release();

          // verify that version of stored procs on
server is correct
          CheckSPVersion();

          // Get IMalloc interface
          hr = CoGetMalloc(1, (LPMALLOC
*)&m_pIMalloc);
```

```
        // Bind parameters for each of the
transactions
        InitNewOrderParams();
        InitPaymentParams();
        InitOrderStatusParams();
        InitDeliveryParams();
        InitStockLevelParams();
}

CTPCC_OLEDB::~CTPCC_OLEDB( void )
{
        if (m_pIMalloc != NULL)
        {
                m_pIMalloc->Release();
        }
        m_pIPaymentCommand->Release();
        m_pIDBCreateCommand->Release();
        m_pIDBCreateSession->Release();

        CoUninitialize();   // uninitialize COM
library
}

/*
*       Check stored procedures version on the
server.
*/
void CTPCC_OLEDB::CheckSPVersion()
{
        HRESULT                     hr;
        char
        db_sp_version[10];
        ICommandText*       pICommandText;
        IAccessor*                  pIAccessor;
        IRowset*            pRowset;
        const ULONG                 nOutputParams
= 1;    // output 1st result set columns
        HACCESSOR
        hTpccVersionOutputAccessor;
        // Structure to bind in accessor
        DBBINDING
        acOutputDBBinding[nOutputParams];
        DBBINDSTATUS
        acOutputDBBindStatus[nOutputParams];
        LONG                        cRows = 1;
        // number of rows returned in the rowset
        ULONG
        cRowsObtained;
        HROW                        rghRow;
        //returned row handles
        HROW*                       prghRow =
&rghRow;


        hr = m_pIDBCreateCommand-
>CreateCommand(NULL, IID_ICommandText, (IUnknown
**)&pICommandText);
        if (FAILED(hr))
        {
                ThrowError(m_pIDBCreateCommand,
COLEDBERR::eCreateCommand, "CheckSPVersion()");
        }

        hr = pICommandText-
>SetCommandText(DBGUID_SQL, L"{call tpcc_version}");
        if (FAILED(hr))
        {
                ThrowError(pICommandText,
COLEDBERR::eSetCommandText, "CheckSPVersion()");
        }

        hr = pICommandText-
>QueryInterface(IID_IAccessor, (void **)&pIAccessor);
        if (FAILED(hr))
        {
                ThrowError(pICommandText,
COLEDBERR::eQueryInterface, "CheckSPVersion()");
        }

        // Now fill the binding information for
result set 1 output columns
        InitBindings(&acOutputDBBinding[0],
nOutputParams, eOutputColumn);

        // Binding for a rowset
        SetBinding(&acOutputDBBinding[0], 0,
sizeof(db_sp_version), DBTYPE_STR);

        hr = pIAccessor->CreateAccessor(
                        DBACCESSOR_ROWDATA,
                        nOutputParams,
                        acOutputDBBinding,
                        sizeof(db_sp_version),

&hTpccVersionOutputAccessor,
                        acOutputDBBindStatus);
        if (FAILED(hr))
        {
                ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor, "CheckSPVersion()");
        }

        hr = pICommandText->Execute(NULL,
IID_IRowset, NULL, NULL, (IUnknown **)&pRowset);
        if (FAILED(hr))
        {
                ThrowError(pICommandText,
COLEDBERR::eExecute, "CheckSPVersion()");
        }

        // Fetch the result row handle(s)
        hr = pRowset->GetNextRows(DB_NULL_HCHAPTER,
0, cRows, &cRowsObtained, &prghRow);
        if (FAILED(hr))
        {
                ThrowError(pICommandText,
COLEDBERR::eGetNextRows, "CheckSPVersion()");
        }

        // Fetch the actual row data by handle
        hr = pRowset->GetData(rghRow,
hTpccVersionOutputAccessor, &db_sp_version);
        if (FAILED(hr))
        {
                ThrowError(pICommandText,
COLEDBERR::eGetData, "CheckSPVersion()");
        }

        // Release row(s)
        hr = pRowset->Release();

        pICommandText->Release();

        // Check the retrieved version
        if (strcmp(db_sp_version,sVersion))
                        throw new
CTPCC_OLEDB_ERR(
CTPCC_OLEDB_ERR::ERR_WRONG_SP_VERSION );
}

void CTPCC_OLEDB::ThrowError( IUnknown*
pObjectWithError, COLEDBERR::ACTION eAction, LPCTSTR
szLocation)
{
        HRESULT
        hr;
        //char
        szState[6];
        char
        szMsg[SQL_MAX_MESSAGE_LENGTH];
        char
        szTmp[6*SQL_MAX_MESSAGE_LENGTH];
        COLEDBERR
        *pOLEDBErr;                                  //
not allocated until needed (maybe never)
        int
                iLen;
        // Interfaces
        IErrorInfo*             pIErrorInfoAll
= NULL;
        IErrorInfo*             pIErrorInfoRecord
= NULL;
        IErrorRecords*          pIErrorRecords
= NULL;
        ISupportErrorInfo*      pISupportErrorInfo
= NULL;
        ISQLServerErrorInfo*
pISQLServerErrorInfo    = NULL;
        ISQLErrorInfo*
        pISQLErrorInfo    = NULL;

        // Information used when cannot get custom
error object
        ERRORINFO
        BasicErrorInfo;
        BSTR
        bstrDescription;
        // Number of error records.
        ULONG                   nRecs;
        ULONG                   nRec;

        // SQL Server error information from
ISQLServerErrorInfo.
        SSERRORINFO*            pSSErrorInfo =
NULL;
        OLECHAR*                pSSErrorStrings =
NULL;

        assert(pObjectWithError != NULL);
```

```
        pOLEDBErr = new COLEDBERR(szLocation);

        pOLEDBErr->m_NativeError = 0;
        pOLEDBErr->m_eAction = eAction;
        pOLEDBErr->m_bDeadLock = FALSE;

        szTmp[0] = 0;

        // Only ask for error information if the
interface supports it.
        // Note: SQLOLEDB provider supports error
interface, so this check is
        // for good style only.
        hr = pObjectWithError-
>QueryInterface(IID_ISupportErrorInfo, (void**)
&pISupportErrorInfo);
        if (FAILED(hr))
        {
                _snprintf(szMsg, sizeof(szMsg),
"SupportErrorInfo interface not supported (hr=0x%X)",
hr);
                pOLEDBErr->m_OLEDBErrStr = new
char[strlen(szMsg)+1];
                strcpy(pOLEDBErr->m_OLEDBErrStr,
szMsg);
                throw pOLEDBErr;
        }
        /*if (FAILED(pISupportErrorInfo-
>InterfaceSupportsErrorInfo(IID_InterfaceWithError)))
        {
                _snprintf(szMsg, sizeof(szMsg),
"InterfaceWithError interface not supported");
                pOLEDBErr->m_OLEDBErrStr = new
char[strlen(szMsg)+1];
                strcpy(pOLEDBErr->m_OLEDBErrStr,
szMsg);
                return;
        }*/

        // Do not test the return of GetErrorInfo.
It can succeed and return
        // a NULL pointer in pIErrorInfoAll. Simply
test the pointer.
        GetErrorInfo(0, &pIErrorInfoAll);

        if (pIErrorInfoAll != NULL)
        {
                // Test to see if it's a valid
OLE DB IErrorInfo interface
                // exposing a list of records.
                if (SUCCEEDED(pIErrorInfoAll-
>QueryInterface(IID_IErrorRecords,      (void**)
&pIErrorRecords)))
                {
                        pIErrorRecords-
>GetRecordCount(&nRecs);

                        // Within each record,
retrieve information from each
                        // of the defined
interfaces.
                        for (nRec = 0; nRec <
nRecs; nRec++)
                        {
                                // Request
the generic SQL error interface.
        pIErrorRecords->GetCustomErrorObject(nRec,

        IID_ISQLErrorInfo, // generic SQL error
interface

        (IUnknown**) &pISQLErrorInfo);

                                if
(pISQLErrorInfo != NULL)
                                {
                                        //
Request SQL Server-specific error interface, not the
generic SQL error interface.
        pISQLErrorInfo->QueryInterface(

        IID_ISQLServerErrorInfo, // SQL Server
error interface

        (void**) &pISQLServerErrorInfo);
                                }

                                // Test to
ensure the reference is valid, then
                                // get error
information from ISQLServerErrorInfo.
                                if
(pISQLServerErrorInfo != NULL)
                                {

        pISQLServerErrorInfo-
>GetErrorInfo(&pSSErrorInfo, &pSSErrorStrings);

                                        //
ISQLServerErrorInfo::GetErrorInfo succeeds
                                        //
even when it has nothing to return. Test the
                                        //
pointers before using.
                                        if
(pSSErrorInfo)
                                        {

        // First, add the error message.

        // Convert Unicode error string to ANSI.

        WideCharToMultiByte(CP_THREAD_ACP, 0,

                pSSErrorInfo->pwszMessage, -1,

                szMsg, sizeof(szMsg),

                NULL, NULL);

        // quit if there isn't enough room to
concatenate error text

        if ( (strlen(szMsg) + 2) > (sizeof(szTmp) -
strlen(szTmp)) )

                break;

        // include line break after first error msg

        if (szTmp[0] != 0)

                strcat( szTmp, "\r\n");

        // concatenate the error record to the
overall error message

        strcat( szTmp, szMsg );

        // Second, add the stored procedure name
and line number, if available.

        if (wcslen(pSSErrorInfo->pwszProcedure)>0)

        {

                // Prefix with a line break

                iLen = sprintf(szMsg,
"\r\nProcedure: ");

                // Convert Unicode error string
to ANSI.

        WideCharToMultiByte(CP_THREAD_ACP, 0,

                        pSSErrorInfo-
>pwszProcedure, -1,

                        &szMsg[iLen],
sizeof(szMsg) - iLen,

                        NULL, NULL);

                // Check if have space to add the
line number.

                // Assume the line number takes
no more than 3 digits.

                if ((strlen(szMsg) + 4)<
sizeof(szMsg))

                {
```

```
          _snprintf(&szMsg[strlen(szMsg)],
sizeof(szMsg),
                                        ":%d",
pSSErrorInfo->wLineNumber);
                    }

                    // quit if there isn't enough
room to concatenate error text
                    if ( (strlen(szMsg) + 2) >
(sizeof(szTmp) - strlen(szTmp)) )
                              break;

                    // concatenate the error record
to the overall error message
                    strcat( szTmp, szMsg );

                    // copy the overall error string
to the exception
                    pOLEDBErr->m_OLEDBErrStr = new
char[strlen(szTmp)+1];
                    strcpy(pOLEDBErr->m_OLEDBErrStr,
szTmp);
          }

          // Third, capture the (first) database
error

          if (pOLEDBErr->m_NativeError == 0 &&
pSSErrorInfo->lNative != 0)
          {
                    pOLEDBErr->m_NativeError =
pSSErrorInfo->lNative;

                    // Check for deadlock error code
and set the deadlock flag
                    if (pSSErrorInfo->lNative ==
1205)
                    {
                              pOLEDBErr->m_bDeadLock
= TRUE;
                    }
          }


          // IMalloc::Free needed to release
references
          // on returned values.
          if (m_pIMalloc != NULL)
          {
                    m_pIMalloc-
>Free(pSSErrorStrings);
                    m_pIMalloc->Free(pSSErrorInfo);
          }
                                                  }


          pISQLServerErrorInfo->Release();
                                        }
                                        else
                                        {
                                                  //
Custom error object is not supported.
                                                  //
Use general OLE-DB error interface.

                                                  //
Get the numeric error code
          pIErrorRecords->GetBasicErrorInfo(nRec,
&BasicErrorInfo);

                                                  if
(pOLEDBErr->m_NativeError == 0)
                                                  {
          // Get the failed call HRESULT code, which
is not really the native error
          pOLEDBErr->m_NativeError =
BasicErrorInfo.hrError;
                                                  }

                                                  //
Try to get the string description of the error.
          pIErrorRecords->GetErrorInfo(nRec,
LOCALE_USER_DEFAULT,
(IErrorInfo**)&pIErrorInfoRecord);

                                                  if
(pIErrorInfoRecord)
                                                  {

          pIErrorInfoRecord-
>GetDescription(&bstrDescription);
          // Convert Unicode error string to ANSI.
          WideCharToMultiByte(CP_THREAD_ACP, 0,
                    bstrDescription, -1,
                    szMsg, sizeof(szMsg),
                    NULL, NULL);

          pOLEDBErr->m_OLEDBErrStr = new
char[strlen(szMsg)+1];
          strcpy(pOLEDBErr->m_OLEDBErrStr, szMsg);

                                                  }
                                        }
                              }         // for()
                    }         // if
(SUCCEEDED(pIErrorInfoAll-
>QueryInterface(IID_IErrorRecords, (void**)
&pIErrorRecords)))
                    else
                    {         // No IErrorRecords
interface supported. Use default IErrorInfo.
                              // Note: SQLOLEDB
supports IErrorRecords, so this check is for good
style only.
                              _snprintf(szMsg,
sizeof(szMsg), "IErrorRecords interface not
supported");
                              pOLEDBErr-
>m_OLEDBErrStr = new char[strlen(szMsg)+1];
                              strcpy(pOLEDBErr-
>m_OLEDBErrStr, szMsg);
                    }

                    pIErrorInfoAll->Release();

          }         // if (pIErrorInfoAll != NULL)
          else
          {
                    // No IErrorInfo interface
supported.
                    // Note: SQLOLEDB supports
IErrorInfo, so this check is for good style only.
                    _snprintf(szMsg, sizeof(szMsg),
"IErrorInfo interface not supported");
                    pOLEDBErr->m_OLEDBErrStr = new
char[strlen(szMsg)+1];
                    strcpy(pOLEDBErr->m_OLEDBErrStr,
szMsg);
          }

          throw pOLEDBErr;
}

/*
*
```

```
*         Create a new command object from the SQL
text passed in.
*
*/
void CTPCC_OLEDB::CreateCommand(wchar_t*
szSQLCommand,                           // I: SQL
query for the command

                              ICommandText**
ppICommandText      // O: returned command object

                              )
{
          HRESULT                                   hr;

          // Create a new command object
          hr = m_pIDBCreateCommand-
>CreateCommand(NULL, IID_ICommandText, (IUnknown
**)ppICommandText);
          if (FAILED(hr))
          {
                    ThrowError(m_pIDBCreateCommand,
COLEDBERR::eCreateCommand,
"CTPCC_OLEDB::CreateCommand");
          }

          // Set command text
          hr = (*ppICommandText)-
>SetCommandText(DBGUID_SQL, szSQLCommand);
          if (FAILED(hr))
          {
                    ThrowError(*ppICommandText,
COLEDBERR::eSetCommandText,
"CTPCC_OLEDB::CreateCommand");
          }

          // Prepare the command
          PrepareCommand(*ppICommandText);
}

/*
*         QueryInterface and Prepare in one function
for simplicity.
*         DEFERRED PREPARE property is set to off to
prepare immediatelly.
*/
void CTPCC_OLEDB::PrepareCommand(ICommandText*
pICommandText)
{
          HRESULT                                   hr;
          ICommandPrepare*    pICommandPrepare;
          ICommandProperties* pICommandProperties;
          DBPROPSET
          rowSetPropSet;
          DBPROP
          rowSetProp;

          // Set the deferred prepare property to
false.
          rowSetProp.dwPropertyID =
SSPROP_DEFERPREPARE;
          memset(&rowSetProp.vValue, 0,
sizeof(rowSetProp.vValue));
          rowSetProp.dwOptions =
DBPROPOPTIONS_REQUIRED;
          rowSetProp.colid = DB_NULLID;

          rowSetPropSet.cProperties = 1;
          rowSetPropSet.guidPropertySet =
DBPROPSET_SQLSERVERROWSET;
          rowSetPropSet.rgProperties = &rowSetProp;

          // Query interface for setting properties
          hr = pICommandText-
>QueryInterface(IID_ICommandProperties, (void
**)&pICommandProperties);
          if (FAILED(hr))
          {
                    ThrowError(pICommandText,
COLEDBERR::eQueryInterface,
"CTPCC_OLEDB::PrepareCommand");
          }

          // Set the property set
          hr = pICommandProperties->SetProperties(1,
&rowSetPropSet);
          if (FAILED(hr))
          {
                    ThrowError(pICommandText,
COLEDBERR::eQueryInterface,
"CTPCC_OLEDB::PrepareCommand");
          }

          // Get interface for preparing commands
          hr = pICommandText-
>QueryInterface(IID_ICommandPrepare, (void
**)&pICommandPrepare);
          if (FAILED(hr))
          {
                    ThrowError(pICommandText,
COLEDBERR::eQueryInterface,
"CTPCC_OLEDB::PrepareCommand");
          }

          // Prepare Payment command
          hr = pICommandPrepare->Prepare(0xFFFFFFFF);
          if (FAILED(hr))
          {
                    ThrowError(pICommandPrepare,
COLEDBERR::ePrepare, "CTPCC_OLEDB::PrepareCommand");
          }
}

/*
*         Initialize fields of an array of bindings
structures.
*         Needs to be called before setting
individual parameter/column bindings.
*/
void CTPCC_OLEDB::InitBindings(DBBINDING*
pDBBindings,        // IO: array of bindings

                    int iCount,
                              // I: number of
elements in the array
                    eBindingType BindingType)     //
I: what the bindings will be used for
(parameters/columns)
{
          int i;

          for(i = 0; i < iCount; i++)
    {
                    pDBBindings[i].iOrdinal = i + 1;
        pDBBindings[i].obLength = 0;
        pDBBindings[i].obStatus = 0;
        pDBBindings[i].pTypeInfo = NULL;
        pDBBindings[i].pObject = NULL;
        pDBBindings[i].pBindExt = NULL;
        pDBBindings[i].dwPart = DBPART_VALUE;

                    switch (BindingType)
                    {
                    case eInputParameter:
                              pDBBindings[i].eParamIO
= DBPARAMIO_INPUT;
                              break;
                    case eOutputParameter:
                              pDBBindings[i].eParamIO
= DBPARAMIO_OUTPUT;
                              break;
                    case eInputOutputParameter:
                              pDBBindings[i].eParamIO
= DBPARAMIO_INPUT | DBPARAMIO_OUTPUT;
                              break;
                    case eOutputColumn:
                              pDBBindings[i].eParamIO
= DBPARAMIO_NOTPARAM;
                              break;
                    default:
                              assert(false);      //
this should never happen
                    }

        pDBBindings[i].dwMemOwner =
DBMEMOWNER_CLIENTOWNED;
        pDBBindings[i].dwFlags = 0;
                    pDBBindings[i].bPrecision = 0;
        pDBBindings[i].bScale = 0;
    }
}

/*
*         Perform binding for one parameter or output
column.
*
*/
void CTPCC_OLEDB::SetBinding(DBBINDING* pDBBinding,
// I: binding row structure

                    size_t obValue,
          // I: parameter (column) offset in the user
buffer

                    size_t cbMaxLen,             //
I: parameter (column) length
```

```
                    DBTYPE wType
          // I: parameter (column) type

                    )
{
          pDBBinding->obValue = (ULONG)obValue;
          pDBBinding->cbMaxLen = (ULONG)cbMaxLen;
          pDBBinding->wType = wType;
}

void CTPCC_OLEDB::InitStockLevelParams()
{
          int
                    i;
          HRESULT
          hr;
          wchar_t
          szName[iMAX_SP_NAME_LEN];
          IAccessor*
          pIAccessor;
    const ULONG
          nInputParams = 3;   // input parameters
          const ULONG
          nOutputParams = 1;  // output 1st result
set columns
          // Structure to bind in accessor
          DBBINDING
          acInputDBBinding[nInputParams];
          DBBINDSTATUS
          acInputDBBindStatus[nInputParams];
          DBBINDING
          acOutputDBBinding[nOutputParams];
          DBBINDSTATUS
          acOutputDBBindStatus[nOutputParams];

          // Set command text
          _snwprintf(szName,
sizeof(szName)/sizeof(szName[0]),
                              L"{call
%stpcc_stocklevel (?,?,?)}", m_szSPPrefix);

          // Create and Prepare a new command object
for StockLevel.
          CreateCommand(szName,
&m_pIStockLevelCommand);

          // Describe the consumer buffer by filling
in the array
    // of DBBINDING structures.  Each binding
associates
    // a single parameter to the consumer's buffer.
          InitBindings(&acInputDBBinding[0],
nInputParams, eInputParameter);

          i = 0;
    // StockLevel parameter 1
          SetBinding(&acInputDBBinding[i++],
offsetof(STOCK_LEVEL_DATA, w_id),
sizeof(m_txn.StockLevel.w_id), DBTYPE_I4);

    // StockLevel parameter 2
          SetBinding(&acInputDBBinding[i++],
offsetof(STOCK_LEVEL_DATA, d_id),
sizeof(m_txn.StockLevel.d_id), DBTYPE_UI1);

          // StockLevel parameter 3
          SetBinding(&acInputDBBinding[i++],
offsetof(STOCK_LEVEL_DATA, threshold),
sizeof(m_txn.StockLevel.threshold), DBTYPE_I2);

          hr = m_pIStockLevelCommand-
>QueryInterface(IID_IAccessor, (void **)&pIAccessor);
          if (FAILED(hr))
          {
                    ThrowError(m_pIStockLevelCommand,
COLEDBERR::eQueryInterface,
"InitStockLevelParams()");
          }

          hr = pIAccessor->CreateAccessor(
                              DBACCESSOR_PARAMETERDATA,
                              nInputParams,
                              acInputDBBinding,
                              sizeof(STOCK_LEVEL_DATA),

&m_hStockLevelInputAccessor,
                              acInputDBBindStatus);
          if (FAILED(hr))
          {
                    ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor,
"InitStockLevelParams()");
          }

          m_StockLevelExecuteParams.cParamSets = 1;
          m_StockLevelExecuteParams.hAccessor =
m_hStockLevelInputAccessor;
          m_StockLevelExecuteParams.pData =
&m_txn.StockLevel;

          // Now fill the binding information for
result set 1 output columns
          InitBindings(&acOutputDBBinding[0],
nOutputParams, eOutputColumn);

          // Binding for a rowset that may return
more than one row.
          i = 0;
          // StockLevel output column 1
          SetBinding(&acOutputDBBinding[i++],
offsetof(STOCK_LEVEL_DATA, low_stock),
sizeof(m_txn.StockLevel.low_stock), DBTYPE_I4);

          hr = pIAccessor->CreateAccessor(
                              DBACCESSOR_ROWDATA |
DBACCESSOR_OPTIMIZED,
                              nOutputParams,
                              acOutputDBBinding,
                              sizeof(STOCK_LEVEL_DATA),

&m_hStockLevelOutputAccessor,
                              acOutputDBBindStatus);
          if (FAILED(hr))
          {
                    ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor,
"InitStockLevelParams()");
          }
}

void CTPCC_OLEDB::StockLevel()
{
          HRESULT                         hr;
          int
          iTryCount = 0;
          IRowset*            pRowset;
          LONG                            cRows = 1;
          // number of rows returned in the rowset
          ULONG
          cRowsObtained;
          HROW                            rghRow;
          //returned row handles
          HROW*                           prghRow =
&rghRow;

          while (TRUE)
          {
                    try
                    {
                              // Execute the prepared
command
                              hr =
m_pIStockLevelCommand->Execute(NULL, IID_IRowset,
&m_StockLevelExecuteParams, NULL,


          (IUnknown **)&pRowset);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIStockLevelCommand,
COLEDBERR::eExecute, "StockLevel()");
                              }

                              // Fetch the result row
handle(s)
                              hr = pRowset-
>GetNextRows(DB_NULL_HCHAPTER, 0, cRows,
&cRowsObtained, &prghRow);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIStockLevelCommand,
COLEDBERR::eGetNextRows, "StockLevel()");
                              }

                              // Fetch the actual row
data by handle
                              hr = pRowset-
>GetData(rghRow, m_hStockLevelOutputAccessor,
&m_txn.StockLevel);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIStockLevelCommand,
COLEDBERR::eGetData, "StockLevel()");
                              }
```

```
                              // Release row(s)
                              hr = pRowset-
>ReleaseRows(cRowsObtained, prghRow, NULL, NULL,
NULL);
                              // Release rowset
                              hr = pRowset-
>Release();


          m_txn.StockLevel.exec_status_code = eOK;

                              break;
                    }
                    catch (COLEDBERR *e)
                    {
                              if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                        throw;

                              // hit deadlock;
backoff for increasingly longer period
                              delete e;
                              Sleep(10 * iTryCount);
                    }
          }

//        if (iTryCount)
//                  throw new
CTPCC_OLEDB_ERR(CTPCC_OLEDB_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

void CTPCC_OLEDB::InitNewOrderParams()
{
          int
                    i, j, iOlCount;
          HRESULT
          hr;
          wchar_t
          szName[iMAX_SP_NAME_LEN];
          IAccessor*
          pIAccessor;
    const ULONG
          nInputParams = 5 +
3*MAX_OL_NEW_ORDER_ITEMS;     // input parameters
          const ULONG
          nOutputParams = 5;  // output 1st result
set columns
          const ULONG
          nOutputParams2 = 8; // output 2nd result
set columns
          // Structure to bind in accessor
          DBBINDING
          acInputDBBinding[nInputParams];
          DBBINDSTATUS
          acInputDBBindStatus[nInputParams];
          DBBINDING
          acOutputDBBinding[nOutputParams];
          DBBINDSTATUS
          acOutputDBBindStatus[nOutputParams];
          DBBINDING
          acOutputDBBinding2[nOutputParams2];
          DBBINDSTATUS
          acOutputDBBindStatus2[nOutputParams2];

          // Describe the consumer buffer by filling
in the array
    // of DBBINDING structures.  Each binding
associates
    // a single parameter to the consumer's buffer.
          InitBindings(&acInputDBBinding[0],
nInputParams, eInputParameter);

          i = 0;
    // NewOrder parameter 1
          SetBinding(&acInputDBBinding[i++],
offsetof(NEW_ORDER_DATA, w_id),
sizeof(m_txn.NewOrder.w_id), DBTYPE_I4);

    // NewOrder parameter 2
          SetBinding(&acInputDBBinding[i++],
offsetof(NEW_ORDER_DATA, d_id),
sizeof(m_txn.NewOrder.d_id), DBTYPE_UI1);

          // NewOrder parameter 3
          SetBinding(&acInputDBBinding[i++],
offsetof(NEW_ORDER_DATA, c_id),
sizeof(m_txn.NewOrder.c_id), DBTYPE_I4);

          // NewOrder parameter 4
          SetBinding(&acInputDBBinding[i++],
offsetof(NEW_ORDER_DATA, o_ol_cnt),
sizeof(m_txn.NewOrder.o_ol_cnt), DBTYPE_UI1);

          // NewOrder parameter 5
          SetBinding(&acInputDBBinding[i++],
offsetof(NEW_ORDER_DATA, o_all_local),
sizeof(m_txn.NewOrder.o_all_local), DBTYPE_UI1);

          for (j=0; j<MAX_OL_NEW_ORDER_ITEMS; j++)
          {

          SetBinding(&acInputDBBinding[i++],
offsetof(NEW_ORDER_DATA, OL[j].ol_i_id),
sizeof(m_txn.NewOrder.OL[j].ol_i_id), DBTYPE_I4);

          SetBinding(&acInputDBBinding[i++],
offsetof(NEW_ORDER_DATA, OL[j].ol_supply_w_id),
sizeof(m_txn.NewOrder.OL[j].ol_supply_w_id),
DBTYPE_I4);

          SetBinding(&acInputDBBinding[i++],
offsetof(NEW_ORDER_DATA, OL[j].ol_quantity),
sizeof(m_txn.NewOrder.OL[j].ol_quantity), DBTYPE_I2);

          }


          // Now fill the binding information for
result set 1 output columns
          InitBindings(&acOutputDBBinding[0],
nOutputParams, eOutputColumn);

          // Binding for the order line rowsets (each
consist of one row).
          // Bind to offsets of the OL_NEW_ORDER_DATA
structure instead of NEW_ORDER_DATA.
          // IRowset::GetData() will be passed
individual array slots OL[i] to fetch the data
          // from the row set.

          i = 0;
          // NewOrder output column 1
          SetBinding(&acOutputDBBinding[i++],
offsetof(OL_NEW_ORDER_DATA, ol_i_name),
sizeof(m_txn.NewOrder.OL[0].ol_i_name), DBTYPE_STR);

          // NewOrder output column 2
          SetBinding(&acOutputDBBinding[i++],
offsetof(OL_NEW_ORDER_DATA, ol_stock),
sizeof(m_txn.NewOrder.OL[0].ol_stock), DBTYPE_I2);

          // NewOrder output column 3
          SetBinding(&acOutputDBBinding[i++],
offsetof(OL_NEW_ORDER_DATA, ol_brand_generic),
sizeof(m_txn.NewOrder.OL[0].ol_brand_generic),
DBTYPE_STR);

          // NewOrder output column 4
          SetBinding(&acOutputDBBinding[i++],
offsetof(OL_NEW_ORDER_DATA, ol_i_price),
sizeof(m_txn.NewOrder.OL[0].ol_i_price), DBTYPE_R8);

          // NewOrder output column 5
          SetBinding(&acOutputDBBinding[i++],
offsetof(OL_NEW_ORDER_DATA, ol_amount),
sizeof(m_txn.NewOrder.OL[0].ol_amount), DBTYPE_R8);



          // Now fill the binding information for
result set 2 output columns
          InitBindings(&acOutputDBBinding2[0],
nOutputParams2, eOutputColumn);

          i = 0;
          // NewOrder output column 1
          SetBinding(&acOutputDBBinding2[i++],
offsetof(NEW_ORDER_DATA, w_tax),
sizeof(m_txn.NewOrder.w_tax), DBTYPE_R8);

          // NewOrder output column 2
          SetBinding(&acOutputDBBinding2[i++],
offsetof(NEW_ORDER_DATA, d_tax),
sizeof(m_txn.NewOrder.d_tax), DBTYPE_R8);

          // NewOrder output column 3
          SetBinding(&acOutputDBBinding2[i++],
offsetof(NEW_ORDER_DATA, o_id),
sizeof(m_txn.NewOrder.o_id), DBTYPE_I4);

          // NewOrder output column 4
          SetBinding(&acOutputDBBinding2[i++],
offsetof(NEW_ORDER_DATA, c_last),
sizeof(m_txn.NewOrder.c_last), DBTYPE_STR);

          // NewOrder output column 5
```

```
          SetBinding(&acOutputDBBinding2[i++],
offsetof(NEW_ORDER_DATA, c_discount),
sizeof(m_txn.NewOrder.c_discount), DBTYPE_R8);

          // NewOrder output column 6
          SetBinding(&acOutputDBBinding2[i++],
offsetof(NEW_ORDER_DATA, c_credit),
sizeof(m_txn.NewOrder.c_credit), DBTYPE_STR);

          // NewOrder output column 7
          SetBinding(&acOutputDBBinding2[i++],
offsetof(NEW_ORDER_DATA, o_entry_d),
sizeof(m_txn.NewOrder.o_entry_d),
DBTYPE_DBTIMESTAMP);

          // NewOrder output column 8
          SetBinding(&acOutputDBBinding2[i++],
offsetof(NEW_ORDER_DATA, o_commit_flag),
sizeof(m_txn.NewOrder.o_commit_flag), DBTYPE_I2);

          for (j=0; j<MAX_OL_NEW_ORDER_ITEMS; j++)
          {
                    // Set command text first

                    // Print the fixed first portion
of parameters
                    i = _snwprintf(szName,
sizeof(szName)/sizeof(szName[0]),

          L"{call %stpcc_neworder (?,?,?,?,?",
m_szSPPrefix);

                    // Now print the variable portion
depending on the number of order line parameters
                    for (iOlCount = 0; iOlCount <= j;
++iOlCount)
                    {
                              i +=
_snwprintf(&szName[i],
sizeof(szName)/sizeof(szName[0]) - i, L",?,?,?");

                    }

                    // Print the fixed end
                    if (j != MAX_OL_NEW_ORDER_ITEMS -
1)
                    {         // append 'default' for
the parameters that are not used
                              i +=
_snwprintf(&szName[i],
sizeof(szName)/sizeof(szName[0]) - i, L",default)}");

                    }
                    else      // using all 15 order
line parameters
                    {
                              i +=
_snwprintf(&szName[i],
sizeof(szName)/sizeof(szName[0]) - i, L")}");
                    }

                    // Create and Prepare a new
command object for NewOrder.
                              CreateCommand(szName,
&m_pINewOrderCommand[j]);

                    // Now create the input accessor
for this prepared command
                    hr = m_pINewOrderCommand[j]-
>QueryInterface(IID_IAccessor, (void **)&pIAccessor);
                    if (FAILED(hr))
                    {

          ThrowError(m_pINewOrderCommand[j],
COLEDBERR::eQueryInterface, "InitNewOrderParams()");
                    }

                    hr = pIAccessor->CreateAccessor(


          DBACCESSOR_PARAMETERDATA,

                                                  5 +
3 * (j + 1),


          acInputDBBinding,


          sizeof(NEW_ORDER_DATA),


          &m_hNewOrderInputAccessor[j],


          acInputDBBindStatus);
                    if (FAILED(hr))
                    {
                              ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor, "InitNewOrderParams()");
                    }



          m_NewOrderExecuteParams[j].cParamSets = 1;
                    //
m_NewOrderExecuteParams.hAccessor is set dynamically
at run-time
                    // based on the number of new
order items for the particular transaction call.

          m_NewOrderExecuteParams[j].hAccessor =
m_hNewOrderInputAccessor[j];
                    m_NewOrderExecuteParams[j].pData
= &m_txn.NewOrder;

                    // Create accessor for the first
rowset
                    hr = pIAccessor->CreateAccessor(
                              DBACCESSOR_ROWDATA |
DBACCESSOR_OPTIMIZED,
                              nOutputParams,
                              acOutputDBBinding,

          sizeof(OL_NEW_ORDER_DATA),

          &m_hNewOrderOutputAccessor[j],
                              acOutputDBBindStatus);
                    if (FAILED(hr))
                    {
                              ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor, "InitNewOrderParams()");
                    }

                    // Create accessor for the second
rowset
                    hr = pIAccessor->CreateAccessor(
                              DBACCESSOR_ROWDATA, //
cannot be optimized too because #1 accessor is
                              nOutputParams2,
                              acOutputDBBinding2,
                              sizeof(NEW_ORDER_DATA),

          &m_hNewOrderOutputAccessor2[j],
                              acOutputDBBindStatus2);
                    if (FAILED(hr))
                    {
                              ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor, "InitNewOrderParams()");
                    }

                    pIAccessor->Release();
          }
}

void CTPCC_OLEDB::NewOrder()
{
          HRESULT                                   hr;
          int
          iTryCount = 0;
          IMultipleResults*   pMultipleResults;
          IRowset*                    pRowset;
          IRowset*                    pRowset2;
          LONG
          cRows = 1;          // number of rows
returned in the 1st rowset
          ULONG
          cRowsObtained;
          HROW
          rghRows;            //returned row handles
for the 1st result set
          HROW*
          prghRows = &rghRows;
          LONG
          cRows2 = 1;         // number of rows
returned in the 2nd rowset
          ULONG
          cRowsObtained2;
          HROW
          rghRows2;           //returned row handle
for the 2nd result set
          HROW*
          prghRows2 = &rghRows2;
          int
          i;
          long
          lRowsAffected;      // the number of
affected rows for a rowset
```

```
        int
        iHandleIndex;       // index into the
handle arrays based on the orders count

        // check whether any order lines are for a
remote warehouse
        m_txn.NewOrder.o_all_local = 1;
        for (i = 0; i < m_txn.NewOrder.o_ol_cnt;
i++)
        {
                if
(m_txn.NewOrder.OL[i].ol_supply_w_id !=
m_txn.NewOrder.w_id)
                {

        m_txn.NewOrder.o_all_local = 0;  // at
least one remote warehouse
                        break;
                }
        }

        iHandleIndex = m_txn.NewOrder.o_ol_cnt - 1;
        // for convenience

        while (TRUE)
        {
                try
                {
                        // Execute the prepared
command (according to the number of new orders)
                        // Ask for
IMultipleResults because it returns 2 rowsets.
                        hr =
m_pINewOrderCommand[iHandleIndex]->Execute(


        NULL, IID_IMultipleResults,


        &m_NewOrderExecuteParams[iHandleIndex],


        NULL,


        (IUnknown **)&pMultipleResults);
                        if (FAILED(hr))
                        {

        ThrowError(m_pINewOrderCommand[iHandleIndex
], COLEDBERR::eExecute, "NewOrder()");
                        }


        //////////////////////////
                        // Get order line
results

        //////////////////////////

        m_txn.NewOrder.total_amount = 0;
                        for (i = 0; i <
m_txn.NewOrder.o_ol_cnt; ++i)
                        {
                                // Get the
first rowset object
                                hr =
pMultipleResults->GetResult(NULL, 0, IID_IRowset,
&lRowsAffected, (IUnknown **)&pRowset);
                                if
(FAILED(hr))
                                {

        char szTmp[256];


        _snprintf(szTmp, sizeof(szTmp), "NewOrder()
result set %d, hr=0x%X", i, hr);


        ThrowError(m_pINewOrderCommand[m_txn.NewOrd
er.o_ol_cnt - 1], COLEDBERR::eGetResult, szTmp);
                                }

                                // Fetch the
result row handle(s)
                                hr = pRowset-
>GetNextRows(DB_NULL_HCHAPTER, 0, cRows,
&cRowsObtained, &prghRows);
                                if
(FAILED(hr))
                                {

        ThrowError(m_pINewOrderCommand[iHandleIndex
], COLEDBERR::eGetNextRows, "NewOrder()");
                                }

                                // Fetch the
actual row data by handle
                                hr = pRowset-
>GetData(rghRows,
m_hNewOrderOutputAccessor[iHandleIndex],
&m_txn.NewOrder.OL[i]);
                                if
(FAILED(hr))
                                {

        ThrowError(m_pINewOrderCommand[iHandleIndex
], COLEDBERR::eGetData, "NewOrder()");
                                }


        m_txn.NewOrder.total_amount +=
m_txn.NewOrder.OL[i].ol_amount;

                                // Release
row(s)
                                hr = pRowset-
>ReleaseRows(cRowsObtained, prghRows, NULL, NULL,
NULL);
                                // Release
rowset
                                hr = pRowset-
>Release();
                        }

        ////////////////////////////////
                        // Get the second
rowset object

        ////////////////////////////////
                        hr = pMultipleResults-
>GetResult(NULL, 0, IID_IRowset, &lRowsAffected,
(IUnknown **)&pRowset2);
                        if (FAILED(hr))
                        {
                                char
szTmp[256];


        _snprintf(szTmp, sizeof(szTmp), "NewOrder()
result set %d, hr=%d", i, hr);


        ThrowError(m_pINewOrderCommand[iHandleIndex
], COLEDBERR::eGetResult, szTmp);
                        }

                        // Fetch the result row
handle(s)
                        hr = pRowset2-
>GetNextRows(DB_NULL_HCHAPTER, 0, cRows2,
&cRowsObtained2, &prghRows2);
                        if (FAILED(hr))
                        {

        ThrowError(m_pINewOrderCommand[iHandleIndex
], COLEDBERR::eGetNextRows, "NewOrder()");
                        }

                        // Fetch the actual row
data by handle
                        hr = pRowset2-
>GetData(rghRows2,
m_hNewOrderOutputAccessor2[iHandleIndex],
&m_txn.NewOrder);
                        if (FAILED(hr))
                        {

        ThrowError(m_pINewOrderCommand[iHandleIndex
], COLEDBERR::eGetData, "NewOrder()");
                        }

                        // Release row(s)
                        hr = pRowset2-
>ReleaseRows(cRowsObtained2, prghRows2, NULL, NULL,
NULL);
                        // Release rowset
                        hr = pRowset2-
>Release();

                        // Release the common
MultipleResults interface
                        hr = pMultipleResults-
>Release();

                        if
(m_txn.NewOrder.o_all_local == 1)
```

```
                                {

          m_txn.NewOrder.total_amount *= ((1 +
m_txn.NewOrder.w_tax + m_txn.NewOrder.d_tax) * (1 -
m_txn.NewOrder.c_discount));

          m_txn.NewOrder.exec_status_code = eOK;
                                }
                                else
                                {

          m_txn.NewOrder.exec_status_code =
eInvalidItem;
                                }

                                break;

                    }
                    catch (COLEDBERR *e)
                    {
                                if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                          throw;

                                // hit deadlock;
backoff for increasingly longer period
                                delete e;
                                Sleep(10 * iTryCount);
                    }
          }

//        if (iTryCount)
//                  throw new
CTPCC_OLEDB_ERR(CTPCC_OLEDB_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

void CTPCC_OLEDB::InitPaymentParams()
{
          int
                    i;
          HRESULT
          hr;
          wchar_t
          szName[iMAX_SP_NAME_LEN];
          IAccessor*
          pIAccessor;
    const ULONG
          nInputParams = 7;   // input parameters
          const ULONG
          nOutputParams = 27; // output result set
columns
          // Structure to bind in accessor
          DBBINDING
          acInputDBBinding[nInputParams];
          DBBINDSTATUS
          acInputDBBindStatus[nInputParams];
          DBBINDING
          acOutputDBBinding[nOutputParams];
          DBBINDSTATUS
          acOutputDBBindStatus[nOutputParams];

          // Set command text
                    _snwprintf(szName,
sizeof(szName)/sizeof(szName[0]), L"{call
%stpcc_payment(?,?,?,?,?,?,?)}", m_szSPPrefix);

                    // Create and Prepare a new command object
for Payment.
                    CreateCommand(szName, &m_pIPaymentCommand);


                    // Describe the consumer buffer by filling
in the array
          // of DBBINDING structures.  Each binding
associates
          // a single parameter to the consumer's buffer.
                    InitBindings(&acInputDBBinding[0],
nInputParams, eInputParameter);

                    i = 0;
          // Payment parameter 1
                    SetBinding(&acInputDBBinding[i++],
offsetof(PAYMENT_DATA, w_id),
sizeof(m_txn.Payment.w_id), DBTYPE_I4);

          // Payment parameter 2
                    SetBinding(&acInputDBBinding[i++],
offsetof(PAYMENT_DATA, c_w_id),
sizeof(m_txn.Payment.c_w_id), DBTYPE_I4);

                    // Payment parameter 3
                    SetBinding(&acInputDBBinding[i++],
offsetof(PAYMENT_DATA, h_amount),
sizeof(m_txn.Payment.h_amount), DBTYPE_R8);

          // Payment parameter 4
                    SetBinding(&acInputDBBinding[i++],
offsetof(PAYMENT_DATA, d_id),
sizeof(m_txn.Payment.d_id), DBTYPE_UI1);

                    // Payment parameter 5
                    SetBinding(&acInputDBBinding[i++],
offsetof(PAYMENT_DATA, c_d_id),
sizeof(m_txn.Payment.c_d_id), DBTYPE_UI1);

                    // Payment parameter 6
                    SetBinding(&acInputDBBinding[i++],
offsetof(PAYMENT_DATA, c_id),
sizeof(m_txn.Payment.c_id), DBTYPE_I4);

                    // Payment parameter 7
                    SetBinding(&acInputDBBinding[i++],
offsetof(PAYMENT_DATA, c_last),
sizeof(m_txn.Payment.c_last), DBTYPE_STR);

                    hr = m_pIPaymentCommand-
>QueryInterface(IID_IAccessor, (void **)&pIAccessor);
                    if (FAILED(hr))
                    {
                              ThrowError(m_pIPaymentCommand,
COLEDBERR::eQueryInterface, "InitPaymentParams()");
                    }

                    hr = pIAccessor->CreateAccessor(
                                        DBACCESSOR_PARAMETERDATA,
                                        nInputParams,
                                        acInputDBBinding,
                                        sizeof(PAYMENT_DATA),
                                        &m_hPaymentInputAccessor,
                                        acInputDBBindStatus);
                    if (FAILED(hr))
                    {
                              ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor, "InitPaymentParams()");
                    }

                    m_PaymentExecuteParams.cParamSets = 1;
                    m_PaymentExecuteParams.hAccessor =
m_hPaymentInputAccessor;
                    m_PaymentExecuteParams.pData =
&m_txn.Payment;

                    // Now fill the binding information for
output columns
                    InitBindings(&acOutputDBBinding[0],
nOutputParams, eOutputColumn);

                    i = 0;
                    // Payment output column 1
                    SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_id),
sizeof(m_txn.Payment.c_id), DBTYPE_I4);

                    // Payment output column 2
                    SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_last),
sizeof(m_txn.Payment.c_last), DBTYPE_STR);

                    // Payment output column 3
                    SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, h_date),
sizeof(m_txn.Payment.h_date), DBTYPE_DBTIMESTAMP);

                    // Payment output column 4
                    SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, w_street_1),
sizeof(m_txn.Payment.w_street_1), DBTYPE_STR);

                    // Payment output column 5
                    SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, w_street_2),
sizeof(m_txn.Payment.w_street_2), DBTYPE_STR);

                    // Payment output column 6
                    SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, w_city),
sizeof(m_txn.Payment.w_city), DBTYPE_STR);

                    // Payment output column 7
                    SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, w_state),
sizeof(m_txn.Payment.w_state), DBTYPE_STR);

                    // Payment output column 8
                    SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, w_zip),
sizeof(m_txn.Payment.w_zip), DBTYPE_STR);
```

```
          // Payment output column 9
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_street_1),
sizeof(m_txn.Payment.d_street_1), DBTYPE_STR);

          // Payment output column 10
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_street_2),
sizeof(m_txn.Payment.d_street_2), DBTYPE_STR);

          // Payment output column 11
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_city),
sizeof(m_txn.Payment.d_city), DBTYPE_STR);

          // Payment output column 12
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_state),
sizeof(m_txn.Payment.d_state), DBTYPE_STR);

          // Payment output column 13
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_zip),
sizeof(m_txn.Payment.d_zip), DBTYPE_STR);

          // Payment output column 14
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_first),
sizeof(m_txn.Payment.c_first), DBTYPE_STR);

          // Payment output column 15
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_middle),
sizeof(m_txn.Payment.c_middle), DBTYPE_STR);

          // Payment output column 16
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_street_1),
sizeof(m_txn.Payment.d_street_1), DBTYPE_STR);

          // Payment output column 17
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_street_2),
sizeof(m_txn.Payment.d_street_2), DBTYPE_STR);

          // Payment output column 18
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_city),
sizeof(m_txn.Payment.d_city), DBTYPE_STR);

          // Payment output column 19
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_state),
sizeof(m_txn.Payment.d_state), DBTYPE_STR);

          // Payment output column 20
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, d_zip),
sizeof(m_txn.Payment.d_zip), DBTYPE_STR);

          // Payment output column 21
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_phone),
sizeof(m_txn.Payment.c_phone), DBTYPE_STR);

          // Payment output column 22
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_since),
sizeof(m_txn.Payment.c_since), DBTYPE_DBTIMESTAMP);

          // Payment output column 23
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_credit),
sizeof(m_txn.Payment.c_credit), DBTYPE_STR);

          // Payment output column 24
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_credit_lim),
sizeof(m_txn.Payment.c_credit_lim), DBTYPE_R8);

          // Payment output column 25
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_discount),
sizeof(m_txn.Payment.c_discount), DBTYPE_R8);

          // Payment output column 26
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_balance),
sizeof(m_txn.Payment.c_balance), DBTYPE_R8);

          // Payment output column 27
          SetBinding(&acOutputDBBinding[i++],
offsetof(PAYMENT_DATA, c_data),
sizeof(m_txn.Payment.c_data), DBTYPE_STR);

          hr = pIAccessor->CreateAccessor(
                              DBACCESSOR_ROWDATA |
DBACCESSOR_OPTIMIZED,
                              nOutputParams,
                              acOutputDBBinding,
                              sizeof(PAYMENT_DATA),

&m_hPaymentOutputAccessor,
                              acOutputDBBindStatus);
          if (FAILED(hr))
          {
                    ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor, "InitPaymentParams()");
          }
}

void CTPCC_OLEDB::Payment()
{
          HRESULT                         hr;
          int
          iTryCount = 0;
          IRowset*            pRowset;
          LONG                            cRows = 1;
          // number of rows returned in the rowset
          ULONG
          cRowsObtained;
          HROW                            rghRow;
          //returned row handles
          HROW*                           prghRow =
&rghRow;


          if (m_txn.Payment.c_id != 0)
                    m_txn.Payment.c_last[0] = 0;

          while (TRUE)
          {
                    try
                    {
                              // Execute the prepared
command
                              hr =
m_pIPaymentCommand->Execute(NULL, IID_IRowset,
&m_PaymentExecuteParams, NULL,


          (IUnknown **)&pRowset);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIPaymentCommand,
COLEDBERR::eExecute, "Payment()");
                              }

                              // Fetch the result row
handle(s)
                              hr = pRowset-
>GetNextRows(DB_NULL_HCHAPTER, 0, cRows,
&cRowsObtained, &prghRow);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIPaymentCommand,
COLEDBERR::eGetNextRows, "Payment()");
                              }

                              // Fetch the actual row
data by handle
                              hr = pRowset-
>GetData(rghRow, m_hPaymentOutputAccessor,
&m_txn.Payment);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIPaymentCommand,
COLEDBERR::eGetData, "Payment()");
                              }

                              // Release row(s)
                              hr = pRowset-
>ReleaseRows(cRowsObtained, prghRow, NULL, NULL,
NULL);
                              // Release rowset
                              hr = pRowset-
>Release();

                              if (m_txn.Payment.c_id
== 0)
                                        throw new
CTPCC_OLEDB_ERR( CTPCC_OLEDB_ERR::ERR_INVALID_CUST );
                              else
```

```
        m_txn.Payment.exec_status_code = eOK;

                        break;
                }
                catch (COLEDBERR *e)
                {
                        if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                throw;

                        // hit deadlock;
backoff for increasingly longer period
                        delete e;
                        Sleep(10 * iTryCount);
                }
        }

//      if (iTryCount)
//              throw new
CTPCC_OLEDB_ERR(CTPCC_OLEDB_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

void CTPCC_OLEDB::InitOrderStatusParams()
{
        int
                i;
        HRESULT
        hr;
        wchar_t
        szName[iMAX_SP_NAME_LEN];
        IAccessor*
        pIAccessor;
    const ULONG
        nInputParams = 4;   // input parameters
        const ULONG
        nOutputParams = 5;  // output 1st result
set columns
        const ULONG
        nOutputParams2 = 8; // output 2nd result
set columns
        // Structure to bind in accessor
        DBBINDING
        acInputDBBinding[nInputParams];
        DBBINDSTATUS
        acInputDBBindStatus[nInputParams];
        DBBINDING
        acOutputDBBinding[nOutputParams];
        DBBINDSTATUS
        acOutputDBBindStatus[nOutputParams];
        DBBINDING
        acOutputDBBinding2[nOutputParams2];
        DBBINDSTATUS
        acOutputDBBindStatus2[nOutputParams2];

        // Set command text
        _snwprintf(szName,
sizeof(szName)/sizeof(szName[0]),
                        L"{call
%stpcc_orderstatus (?,?,?,?)}", m_szSPPrefix);
```

```
        // Create and Prepare a new command object
for OrderStatus.
        CreateCommand(szName,
&m_pIOrderStatusCommand);

        // Describe the consumer buffer by filling
in the array
    // of DBBINDING structures.  Each binding
associates
    // a single parameter to the consumer's buffer.
        InitBindings(&acInputDBBinding[0],
nInputParams, eInputParameter);

        i = 0;
    // OrderStatus parameter 1
        SetBinding(&acInputDBBinding[i++],
offsetof(ORDER_STATUS_DATA, w_id),
sizeof(m_txn.OrderStatus.w_id), DBTYPE_I4);

    // OrderStatus parameter 2
        SetBinding(&acInputDBBinding[i++],
offsetof(ORDER_STATUS_DATA, d_id),
sizeof(m_txn.OrderStatus.d_id), DBTYPE_UI1);

        // OrderStatus parameter 3
        SetBinding(&acInputDBBinding[i++],
offsetof(ORDER_STATUS_DATA, c_id),
sizeof(m_txn.OrderStatus.c_id), DBTYPE_I4);

        // OrderStatus parameter 4
        SetBinding(&acInputDBBinding[i++],
offsetof(ORDER_STATUS_DATA, c_last),
sizeof(m_txn.OrderStatus.c_last), DBTYPE_STR);

        hr = m_pIOrderStatusCommand-
>QueryInterface(IID_IAccessor, (void **)&pIAccessor);
        if (FAILED(hr))
        {

        ThrowError(m_pIOrderStatusCommand,
COLEDBERR::eQueryInterface,
"InitOrderStatusParams()");
        }

        hr = pIAccessor->CreateAccessor(
                        DBACCESSOR_PARAMETERDATA,
                        nInputParams,
                        acInputDBBinding,

sizeof(ORDER_STATUS_DATA),

&m_hOrderStatusInputAccessor,
                        acInputDBBindStatus);
        if (FAILED(hr))
        {
                ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor,
"InitOrderStatusParams()");
        }

        m_OrderStatusExecuteParams.cParamSets = 1;
        m_OrderStatusExecuteParams.hAccessor =
m_hOrderStatusInputAccessor;
```

```
        m_OrderStatusExecuteParams.pData =
&m_txn.OrderStatus;

        // Now fill the binding information for
result set 1 output columns
        InitBindings(&acOutputDBBinding[0],
nOutputParams, eOutputColumn);

        // Binding for a rowset that may return
more than one row.
        // Bind to offsets of the
OL_ORDER_STATUS_DATA structure instead of
ORDER_STATUS_DATA.
        // IRowset::GetData() will be passed
individual array slots OL[i] to fetch the data
        // from the row set.

        i = 0;
        // OrderStatus output column 1
        SetBinding(&acOutputDBBinding[i++],
offsetof(OL_ORDER_STATUS_DATA, ol_supply_w_id),
sizeof(m_txn.OrderStatus.OL[0].ol_supply_w_id),
DBTYPE_I4);

        // OrderStatus output column 2
        SetBinding(&acOutputDBBinding[i++],
offsetof(OL_ORDER_STATUS_DATA, ol_i_id),
sizeof(m_txn.OrderStatus.OL[0].ol_i_id), DBTYPE_I4);

        // OrderStatus output column 3
        SetBinding(&acOutputDBBinding[i++],
offsetof(OL_ORDER_STATUS_DATA, ol_quantity),
sizeof(m_txn.OrderStatus.OL[0].ol_quantity),
DBTYPE_I2);

        // OrderStatus output column 4
        SetBinding(&acOutputDBBinding[i++],
offsetof(OL_ORDER_STATUS_DATA, ol_amount),
sizeof(m_txn.OrderStatus.OL[0].ol_amount),
DBTYPE_R8);

        // OrderStatus output column 5
        SetBinding(&acOutputDBBinding[i++],
offsetof(OL_ORDER_STATUS_DATA, ol_delivery_d),
sizeof(m_txn.OrderStatus.OL[0].ol_delivery_d),
DBTYPE_DBTIMESTAMP);

        hr = pIAccessor->CreateAccessor(
                        DBACCESSOR_ROWDATA |
DBACCESSOR_OPTIMIZED,
                        nOutputParams,
                        acOutputDBBinding,

sizeof(OL_ORDER_STATUS_DATA),

&m_hOrderStatusOutputAccessor,
                        acOutputDBBindStatus);
        if (FAILED(hr))
        {
                ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor,
"InitOrderStatusParams()");
        }
```

```
        // Now fill the binding information for
result set 2 output columns
        InitBindings(&acOutputDBBinding2[0],
nOutputParams2, eOutputColumn);

        i = 0;
        // OrderStatus output column 1
        SetBinding(&acOutputDBBinding2[i++],
offsetof(ORDER_STATUS_DATA, c_id),
sizeof(m_txn.OrderStatus.c_id), DBTYPE_I4);

        // OrderStatus output column 2
        SetBinding(&acOutputDBBinding2[i++],
offsetof(ORDER_STATUS_DATA, c_last),
sizeof(m_txn.OrderStatus.c_last), DBTYPE_STR);

        // OrderStatus output column 3
        SetBinding(&acOutputDBBinding2[i++],
offsetof(ORDER_STATUS_DATA, c_first),
sizeof(m_txn.OrderStatus.c_first), DBTYPE_STR);

        // OrderStatus output column 4
        SetBinding(&acOutputDBBinding2[i++],
offsetof(ORDER_STATUS_DATA, c_middle),
sizeof(m_txn.OrderStatus.c_middle), DBTYPE_STR);

        // OrderStatus output column 5
        SetBinding(&acOutputDBBinding2[i++],
offsetof(ORDER_STATUS_DATA, o_entry_d),
sizeof(m_txn.OrderStatus.o_entry_d),
DBTYPE_DBTIMESTAMP);

        // OrderStatus output column 7
        SetBinding(&acOutputDBBinding2[i++],
offsetof(ORDER_STATUS_DATA, o_carrier_id),
sizeof(m_txn.OrderStatus.o_carrier_id), DBTYPE_I2);

        // OrderStatus output column 8
        SetBinding(&acOutputDBBinding2[i++],
offsetof(ORDER_STATUS_DATA, c_balance),
sizeof(m_txn.OrderStatus.c_balance), DBTYPE_R8);

        // OrderStatus output column 9
        SetBinding(&acOutputDBBinding2[i++],
offsetof(ORDER_STATUS_DATA, o_id),
sizeof(m_txn.OrderStatus.o_id), DBTYPE_I4);

        hr = pIAccessor->CreateAccessor(
                        DBACCESSOR_ROWDATA, //
cannot be optimized too because #1 accessor is
                        nOutputParams2,
                        acOutputDBBinding2,
                        sizeof(NEW_ORDER_DATA),

&m_hOrderStatusOutputAccessor2,
                        acOutputDBBindStatus2);
        if (FAILED(hr))
        {
                ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor,
"InitOrderStatusParams()");
        }
}

void CTPCC_OLEDB::OrderStatus()
{
        HRESULT                                 hr;
        int
        iTryCount = 0;
        IMultipleResults*   pMultipleResults;
        IRowset*                        pRowset;
        IRowset*                        pRowset2;
        LONG
        cRows = MAX_OL_ORDER_STATUS_ITEMS;      //
number of rows returned in the 1st rowset
        ULONG
        cRowsObtained;
        HROW
        rghRows[MAX_OL_ORDER_STATUS_ITEMS];
        //returned row handles for the 1st result
set
        HROW*
        prghRows = &rghRows[0];
        LONG
        cRows2 = 1;         // number of rows
returned in the 2nd rowset
        ULONG
        cRowsObtained2;
        HROW
        rghRows2;           //returned row handle
for the 2nd result set
        HROW*
        prghRows2 = &rghRows2;
        int
        i;
        long
        lRowsAffected;      // the number of
affected rows for a rowset

        if (m_txn.OrderStatus.c_id != 0)
                m_txn.OrderStatus.c_last[0] = 0;

        while (TRUE)
        {
                try
                {
                        // Execute the prepared
command
                        // Ask for
IMultipleResults because it returns 2 rowsets.
                        hr =
m_pIOrderStatusCommand->Execute(NULL,
IID_IMultipleResults, &m_OrderStatusExecuteParams,
NULL,


        (IUnknown **)&pMultipleResults);
                        if (FAILED(hr))
                        {

        ThrowError(m_pIOrderStatusCommand,
COLEDBERR::eExecute, "OrderStatus()");
                        }

        //////////////////////////
                        // Get order line
results
        //////////////////////////


                        // Get the first rowset
object
                        hr = pMultipleResults-
>GetResult(NULL, 0, IID_IRowset, &lRowsAffected,
(IUnknown **)&pRowset);
                        if (FAILED(hr))
                        {

        ThrowError(m_pIOrderStatusCommand,
COLEDBERR::eGetResult, "OrderStatus()");
                        }

                        // Fetch the result row
handle(s)
                        hr = pRowset-
>GetNextRows(DB_NULL_HCHAPTER, 0, cRows,
&cRowsObtained, &prghRows);
                        if (FAILED(hr))
                        {

        ThrowError(m_pIOrderStatusCommand,
COLEDBERR::eGetNextRows, "OrderStatus()");
                        }


        m_txn.OrderStatus.o_ol_cnt =
(short)cRowsObtained;

                        // Get the data from
multiple rows in this rowset
                        for (i = 0; i <
m_txn.OrderStatus.o_ol_cnt; ++i)
                        {
                                // Fetch the
actual row data by handle
                                hr = pRowset-
>GetData(rghRows[i], m_hOrderStatusOutputAccessor,
&m_txn.OrderStatus.OL[i]);
                                if
(FAILED(hr))
                                {

        ThrowError(m_pIOrderStatusCommand,
COLEDBERR::eGetData, "OrderStatus()");
                                }
                        }

                        // Release row(s)
                        hr = pRowset-
>ReleaseRows(cRowsObtained, prghRows, NULL, NULL,
NULL);
                        // Release rowset
                        hr = pRowset-
>Release();
```

```
        ////////////////////////////////
                            // Get the second
rowset object

        ////////////////////////////////
                            if
(m_txn.OrderStatus.o_ol_cnt > 0)
                            {
                                    hr =
pMultipleResults->GetResult(NULL, 0, IID_IRowset,
&lRowsAffected, (IUnknown **)&pRowset2);
                                    if
(FAILED(hr))
                                    {

          ThrowError(m_pIOrderStatusCommand,
COLEDBERR::eGetResult, "OrderStatus()");
                                    }

                                    // Fetch the
result row handle(s)
                                    hr =
pRowset2->GetNextRows(DB_NULL_HCHAPTER, 0, cRows2,
&cRowsObtained2, &prghRows2);
                                    if
(FAILED(hr))
                                    {

          ThrowError(m_pIOrderStatusCommand,
COLEDBERR::eGetNextRows, "OrderStatus()");
                                    }

                                    // Fetch the
actual row data by handle
                                    hr =
pRowset2->GetData(rghRows2,
m_hOrderStatusOutputAccessor2, &m_txn.OrderStatus);
                                    if
(FAILED(hr))
                                    {

          ThrowError(m_pIOrderStatusCommand,
COLEDBERR::eGetData, "OrderStatus()");
                                    }

                                    // Release
row(s)
                                    hr =
pRowset2->Release();
                            }

                            // Release the common
MultipleResults interface
                            hr = pMultipleResults-
>Release();

                            if
(m_txn.OrderStatus.o_ol_cnt == 0)
                                    throw new
CTPCC_OLEDB_ERR( CTPCC_OLEDB_ERR::ERR_NO_SUCH_ORDER
);
                                    else if
(m_txn.OrderStatus.c_id == 0 &&
m_txn.OrderStatus.c_last[0] == 0)
                                            throw new
CTPCC_OLEDB_ERR( CTPCC_OLEDB_ERR::ERR_INVALID_CUST );
                                    else

          m_txn.OrderStatus.exec_status_code = eOK;

                                    break;

                    }
                    catch (COLEDBERR *e)
                    {
                                    if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                            throw;

                                    // hit deadlock;
backoff for increasingly longer period
                                    delete e;
                                    Sleep(10 * iTryCount);
                    }
          }

//        if (iTryCount)
//                  throw new
CTPCC_OLEDB_ERR(CTPCC_OLEDB_ERR::ERR_RETRIED_TRANS,
iTryCount);
}

void CTPCC_OLEDB::InitDeliveryParams()
{
          int
                    i;
          HRESULT
          hr;
          wchar_t
          szName[iMAX_SP_NAME_LEN];
          IAccessor*
          pIAccessor;
    const ULONG
          nInputParams = 2;   // input parameters
          const ULONG
          nOutputParams = 10; // output 1st result
set columns
          // Structure to bind in accessor
          DBBINDING
          acInputDBBinding[nInputParams];
          DBBINDSTATUS
          acInputDBBindStatus[nInputParams];
          DBBINDING
          acOutputDBBinding[nOutputParams];
          DBBINDSTATUS
          acOutputDBBindStatus[nOutputParams];

          // Set command text
          _snwprintf(szName,
sizeof(szName)/sizeof(szName[0]),
                                    L"{call %stpcc_delivery
(?,?)}", m_szSPPrefix);

          // Create and Prepare a new command object
for Delivery.
          CreateCommand(szName,
&m_pIDeliveryCommand);

          // Describe the consumer buffer by filling
in the array
    // of DBBINDING structures.  Each binding
associates
    // a single parameter to the consumer's buffer.
          InitBindings(&acInputDBBinding[0],
nInputParams, eInputParameter);

          i = 0;
    // Delivery parameter 1
          SetBinding(&acInputDBBinding[i++],
offsetof(DELIVERY_DATA, w_id),
sizeof(m_txn.Delivery.w_id), DBTYPE_I4);

    // Delivery parameter 2
          SetBinding(&acInputDBBinding[i++],
offsetof(DELIVERY_DATA, o_carrier_id),
sizeof(m_txn.Delivery.o_carrier_id), DBTYPE_I2);

          hr = m_pIDeliveryCommand-
>QueryInterface(IID_IAccessor, (void **)&pIAccessor);
          if (FAILED(hr))
          {
                    ThrowError(m_pIDeliveryCommand,
COLEDBERR::eQueryInterface, "InitDeliveryParams()");
          }

          hr = pIAccessor->CreateAccessor(
                              DBACCESSOR_PARAMETERDATA,
                              nInputParams,
                              acInputDBBinding,
                              sizeof(DELIVERY_DATA),

&m_hDeliveryInputAccessor,
                              acInputDBBindStatus);
          if (FAILED(hr))
          {
                    ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor, "InitDeliveryParams()");
          }

          m_DeliveryExecuteParams.cParamSets = 1;
          m_DeliveryExecuteParams.hAccessor =
m_hDeliveryInputAccessor;
          m_DeliveryExecuteParams.pData =
&m_txn.Delivery;

          // Now fill the binding information for
result set 1 output columns
          InitBindings(&acOutputDBBinding[0],
nOutputParams, eOutputColumn);

          // Binding for a rowset that may return
more than one row.
          for (i = 0; i < 10; ++i)
          {
                    // Delivery output column 1
```

```
                    SetBinding(&acOutputDBBinding[i],
offsetof(DELIVERY_DATA, o_id[i]),
sizeof(m_txn.Delivery.o_id[i]), DBTYPE_I4);
          }

          hr = pIAccessor->CreateAccessor(
                              DBACCESSOR_ROWDATA |
DBACCESSOR_OPTIMIZED,
                              nOutputParams,
                              acOutputDBBinding,
                              sizeof(DELIVERY_DATA),

&m_hDeliveryOutputAccessor,
                              acOutputDBBindStatus);
          if (FAILED(hr))
          {
                    ThrowError(pIAccessor,
COLEDBERR::eCreateAccessor, "InitDeliveryParams()");
          }
}

void CTPCC_OLEDB::Delivery()
{
          HRESULT                         hr;
          int
          iTryCount = 0;
          IRowset*            pRowset;
          LONG                            cRows = 1;
          // number of rows returned in the rowset
          ULONG
          cRowsObtained;
          HROW                            rghRow;
          //returned row handles
          HROW*                           prghRow =
&rghRow;

          while (TRUE)
          {
                    try
                    {
                              // Execute the prepared
command
                              hr =
m_pIDeliveryCommand->Execute(NULL, IID_IRowset,
&m_DeliveryExecuteParams, NULL,


          (IUnknown **)&pRowset);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIDeliveryCommand,
COLEDBERR::eExecute, "Delivery()");
                              }

                              // Fetch the result row
handle(s)
                              hr = pRowset-
>GetNextRows(DB_NULL_HCHAPTER, 0, cRows,
&cRowsObtained, &prghRow);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIDeliveryCommand,
COLEDBERR::eGetNextRows, "Delivery()");
                              }

                              // Fetch the actual row
data by handle
                              hr = pRowset-
>GetData(rghRow, m_hDeliveryOutputAccessor,
&m_txn.Delivery);
                              if (FAILED(hr))
                              {

          ThrowError(m_pIDeliveryCommand,
COLEDBERR::eGetData, "Delivery()");
                              }

                              // Release row(s)
                              hr = pRowset-
>ReleaseRows(cRowsObtained, prghRow, NULL, NULL,
NULL);
                              // Release rowset
                              hr = pRowset-
>Release();


          m_txn.Delivery.exec_status_code = eOK;

                              break;
                    }
                    catch (COLEDBERR *e)
                    {
                              if ((!e->m_bDeadLock)
|| (++iTryCount > iMaxRetries))
                                        throw;

                              // hit deadlock;
backoff for increasingly longer period
                              delete e;
                              Sleep(10 * iTryCount);
                    }
          }

//        if (iTryCount)
//                  throw new
CTPCC_OLEDB_ERR(CTPCC_OLEDB_ERR::ERR_RETRIED_TRANS,
iTryCount);
}
```

tpcc_oledb.h

```
/*        FILE:               TPCC_OLEDB.H
 *                                  Microsoft
TPC-C Kit Ver. 4.20.000
 *                                  Copyright
Microsoft, 1999-2004
 *                                  Written by
Sergey Vasilevskiy
 *             All Rights Reserved
 *
 *
 *        PURPOSE:  Header file for TPC-C txn class
OLE DB implementation.
 *
 *
 */
#pragma once

// need to declare functions for import, unless
define has already been created
// by the DLL's .cpp module for export.
#ifndef DllDecl
#define DllDecl  __declspec( dllimport )
#endif

#define iMAX_SP_NAME_LEN 256  //maximum length of a
stored procedure name with parameters

// Type of parameter and result set column bindings.
enum eBindingType
{
          eInputParameter,
          eOutputParameter,
          eInputOutputParameter,
          eOutputColumn
};

class COLEDBERR : public CBaseErr
{
          public:
                    enum ACTION
                    {
                              eNone,
                              eUnknown,
                              eQueryInterface,
                    // error from QueryInterface
                              eCreateSession,
                              eCreateCommand,
                              eSetCommandText,
                              eExecute,
          // = 6
                              eCreateAccessor,
                              ePrepare,
                              eGetNextRows,
                              eGetData,
                              eGetResult
                    // = 11
                    };

                    COLEDBERR(LPCTSTR szLoc)
                              : CBaseErr(szLoc)
                    {
                              m_eAction = eNone;
                              m_NativeError = 0;
                              m_bDeadLock = FALSE;
                              m_OLEDBErrStr = NULL;
                    };

                    ~COLEDBERR()
                    {
                              if (m_OLEDBErrStr !=
NULL)
```

```
                                        delete []
m_OLEDBErrStr;
                    };

                    ACTION    m_eAction;
                    int
          m_NativeError;
                    BOOL      m_bDeadLock;
                    char   *m_OLEDBErrStr;

                    int                 ErrorType()
{return ERR_TYPE_OLEDB;};
                    char*     ErrorTypeStr() { return
"OLEDB"; }
                    int                 ErrorNum()
{return m_NativeError;};
                    char*     ErrorText() {return
m_OLEDBErrStr;};
                    int                 ErrorAction()
{ return (int)m_eAction; }

};

class CTPCC_OLEDB_ERR : public CBaseErr
{
          public:
                    enum TPCC_OLEDB_ERRS
                    {
                              ERR_WRONG_SP_VERSION =
1,        // "Wrong version of stored procs on
database server"
                              ERR_INVALID_CUST,
                    // "Invalid Customer id,name."
                              ERR_NO_SUCH_ORDER,
                    // "No orders found for
customer."
                              ERR_RETRIED_TRANS,
                    // "Retries before transaction
succeeded."
                    };

                    CTPCC_OLEDB_ERR( int iErr ) {
m_errno = iErr; m_iTryCount = 0; };

                    CTPCC_OLEDB_ERR( int iErr, int
iTryCount ) { m_errno = iErr; m_iTryCount =
iTryCount; };

                    int                 m_errno;
                    int                 m_iTryCount;

                    int                 ErrorType()
{return ERR_TYPE_TPCC_OLEDB;};
                    char*     ErrorTypeStr() { return
"TPCC OLEDB"; }
                    int                 ErrorNum()
{return m_errno;};

                    char*     ErrorText();
};

class DllDecl CTPCC_OLEDB : public CTPCC_BASE
{
          private:
                    // declare variables and private
functions here...
                    BOOL
                    m_bDeadlock;                //
transaction was selected as deadlock victim
                    int
                              m_MaxRetries;
          // retry count on deadlock

                    DBPROPSET
          m_rgInitPropSet;              //
initialization property set used to establish a
connection
                    DBPROP
                    m_InitProperties[4];          //
individual initialization properties

                    IDBCreateSession*
          m_pIDBCreateSession;          // session
(connection) interface
                    IDBCreateCommand*
          m_pIDBCreateCommand;          // SQL
command creation interface

                    IMalloc*
          m_pIMalloc;
          // Needed to release error strings.

                    // StockLevel
                    ICommandText*
          m_pIStockLevelCommand;
                    HACCESSOR
          m_hStockLevelInputAccessor;   // accessor
to bind input parameters
                    HACCESSOR
          m_hStockLevelOutputAccessor;  // accessor
to bind output columns
                    DBPARAMS
          m_StockLevelExecuteParams;              //
parameter structure for Execute

                    // NewOrder
                    // One prepared command for each
possible number of new order line items
                    ICommandText*
          m_pINewOrderCommand[MAX_OL_NEW_ORDER_ITEMS]
;
                    // accessors to bind input
parameters
                    // one for each possible number
of new order line items
                    HACCESSOR
          m_hNewOrderInputAccessor[MAX_OL_NEW_ORDER_I
TEMS];
                    // accessor to bind output
columns of the first rowset
                    HACCESSOR
          m_hNewOrderOutputAccessor[MAX_OL_NEW_ORDER_
ITEMS];
                    // accessor to bind output
columns of the second rowset
                    HACCESSOR
          m_hNewOrderOutputAccessor2[MAX_OL_NEW_ORDER
_ITEMS];
                    // parameter structure for
Execute
                    DBPARAMS
          m_NewOrderExecuteParams[MAX_OL_NEW_ORDER_IT
EMS];

                    // Payment
                    ICommandText*
          m_pIPaymentCommand;
                    HACCESSOR
          m_hPaymentInputAccessor;      // accessor
to bind input parameters
                    HACCESSOR
          m_hPaymentOutputAccessor;     // accessor
to bind output columns
                    DBPARAMS
          m_PaymentExecuteParams;                 //
parameter structure for Execute

                    // OrderStatus
                    ICommandText*
          m_pIOrderStatusCommand;
                    HACCESSOR
          m_hOrderStatusInputAccessor;  // accessor
to bind input parameters
                    HACCESSOR
          m_hOrderStatusOutputAccessor; // accessor
to bind output columns
                    HACCESSOR
          m_hOrderStatusOutputAccessor2;          //
accessor to bind output columns
                    DBPARAMS
          m_OrderStatusExecuteParams;             //
parameter structure for Execute

                    // Delivery
                    ICommandText*
          m_pIDeliveryCommand;
                    HACCESSOR
          m_hDeliveryInputAccessor;     // accessor
to bind input parameters
                    HACCESSOR
          m_hDeliveryOutputAccessor;    // accessor
to bind output columns
                    DBPARAMS
          m_DeliveryExecuteParams;      // parameter
structure for Execute

                    wchar_t
                    m_szSPPrefix[32];   // stored
procedures prefix

                    // new-order specific fields

                    int
                              m_no_commit_flag;

                    void ThrowError( IUnknown*
pObjectWithError, COLEDBERR::ACTION eAction, LPCTSTR
szLocation );
```

```
                    void CheckSPVersion();

                    void InitNewOrderParams();
                    void InitPaymentParams();
                    void InitDeliveryParams();
                    void InitStockLevelParams();
                    void InitOrderStatusParams();

                    // Helper function to create and
prepare a command
                    void CreateCommand(wchar_t*
szSQLCommand, ICommandText** ppICommandText);
                    // Helper function to prepare a
command
                    void PrepareCommand(ICommandText*
pICommand);

                    // Helper function to fill one
binding
                    // Used for both input parameter
and output column bindings
                    void SetBinding(DBBINDING*
pDBBinding, size_t obValue, size_t cbMaxLen, DBTYPE
wType);

                    // Helper function to initialize
an array of bindings
                    void InitBindings(DBBINDING*
pDBBindings, int iCount, eBindingType BindingType);

                    union
                    {
                              NEW_ORDER_DATA
          NewOrder;
                              PAYMENT_DATA
          Payment;
                              DELIVERY_DATA
          Delivery;
                              STOCK_LEVEL_DATA
          StockLevel;
                              ORDER_STATUS_DATA
          OrderStatus;
                    }
          m_txn;

          public:
                    CTPCC_OLEDB(LPCSTR szServer,
LPCSTR szUser, LPCSTR szPassword, LPCSTR szHost,
LPCSTR szDatabase, LPCWSTR szSPPrefix);
                    ~CTPCC_OLEDB(void);

                    inline PNEW_ORDER_DATA
          BuffAddr_NewOrder()           { return
&m_txn.NewOrder;    };
                    inline PPAYMENT_DATA
          BuffAddr_Payment()            { return
&m_txn.Payment;     };
                    inline PDELIVERY_DATA
          BuffAddr_Delivery()           { return
&m_txn.Delivery;    };
                    inline PSTOCK_LEVEL_DATA
          BuffAddr_StockLevel()         { return
&m_txn.StockLevel;  };
                    inline PORDER_STATUS_DATA
          BuffAddr_OrderStatus()        { return
&m_txn.OrderStatus; };

                    void NewOrder                 ();
                    void Payment                  ();
                    void Delivery                 ();
                    void StockLevel               ();
                    void OrderStatus    ();

};


// wrapper routine for class constructor
extern "C" DllDecl CTPCC_OLEDB* CTPCC_OLEDB_new
          ( LPCSTR szServer, LPCSTR szUser, LPCSTR
szPassword, LPCSTR szHost, LPCSTR szDatabase, LPCWSTR
szSPPrefix );

typedef CTPCC_OLEDB* (TYPE_CTPCC_OLEDB)(LPCSTR,
LPCSTR, LPCSTR, LPCSTR, LPCSTR, LPCWSTR);
```

trans.h

```
/*        FILE:               TRANS.H
 *                                        Microsoft
TPC-C Kit Ver. 4.42.000
 *                                        Copyright
Microsoft, 2002
 *              All Rights Reserved
 *
 *                                        Version
4.10.000 audited by Richard Gimarc, Performance
Metrics, 3/17/99
 *
 *        PURPOSE:  Header file for TPC-C structure
templates.
 *
 *  Change history:
 *                  4.42.000 - changed w_id fields
from short to long to support >32K warehouses
 *                  4.20.000 - updated rev number to
match kit
 *                  4.69.000 - updated rev number to
match kit
 */
#pragma once

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

// TIMESTAMP_STRUCT is provided by the ODBC header
file sqltypes.h, but is not available
// when compiling with dblib, so redefined here.
Note: we are using the symbol "__SQLTYPES"
// (declared in sqltypes.h) as a way to determine if
TIMESTAMP_STRUCT has been declared.
#ifndef __SQLTYPES
          typedef struct
          {
                              short
          /* SQLSMALLINT */   year;
                              unsigned short      /*
SQLUSMALLINT */     month;
                              unsigned short      /*
SQLUSMALLINT */     day;
                              unsigned short      /*
SQLUSMALLINT */     hour;
                              unsigned short      /*
SQLUSMALLINT */     minute;
                              unsigned short      /*
SQLUSMALLINT */     second;
                              unsigned long       /*
SQLUINTEGER */      fraction;
          } TIMESTAMP_STRUCT;
#endif

// possible values for exec_status_code after
transaction completes
enum EXEC_STATUS
{
          eOK,                          // 0
          "Transaction commited."
          eInvalidItem,       // 1      "Item number
is not valid."
          eDeliveryFailed     // 2      "Delivery
Post Failed."
};

// transaction structures
typedef struct
{
          // input params
```

```
        long
        ol_supply_w_id;
        long
        ol_i_id;
        short
        ol_quantity;

        // output params
        char
        ol_i_name[I_NAME_LEN+1];
        char
        ol_brand_generic[BRAND_LEN+1];
        double
        ol_i_price;
        double
        ol_amount;
        short
        ol_stock;
} OL_NEW_ORDER_DATA;

typedef struct
{
        // input params
        long                w_id;
        short               d_id;
        long                c_id;
        short               o_ol_cnt;

        // output params
        EXEC_STATUS
        exec_status_code;
        char
        c_last[LAST_NAME_LEN+1];
        char                c_credit[CREDIT_LEN+1];
        double              c_discount;
        double              w_tax;
        double              d_tax;
        long                            o_id;
        short
        o_commit_flag;
        TIMESTAMP_STRUCT    o_entry_d;
        short               o_all_local;
        double              total_amount;
        OL_NEW_ORDER_DATA
        OL[MAX_OL_NEW_ORDER_ITEMS];
} NEW_ORDER_DATA, *PNEW_ORDER_DATA;

typedef struct
{       // input params
        long
w_id;
        short
d_id;
        long
c_id;
        short
c_d_id;
        long
c_w_id;
        double
h_amount;
        char
c_last[LAST_NAME_LEN+1];

        // output params
        EXEC_STATUS
exec_status_code;
        TIMESTAMP_STRUCT     h_date;
        char
w_street_1[ADDRESS_LEN+1];
        char
w_street_2[ADDRESS_LEN+1];
        char
w_city[ADDRESS_LEN+1];
        char
w_state[STATE_LEN+1];
        char
w_zip[ZIP_LEN+1];
        char
d_street_1[ADDRESS_LEN+1];
        char
d_street_2[ADDRESS_LEN+1];
        char
d_city[ADDRESS_LEN+1];
        char
d_state[STATE_LEN+1];
        char
d_zip[ZIP_LEN+1];
        char
c_first[FIRST_NAME_LEN+1];
        char
c_middle[MIDDLE_NAME_LEN + 1];
        char
c_street_1[ADDRESS_LEN+1];
        char
c_street_2[ADDRESS_LEN+1];
        char
c_city[ADDRESS_LEN+1];
        char
c_state[STATE_LEN+1];
        char
c_zip[ZIP_LEN+1];
        char
c_phone[PHONE_LEN+1];
        TIMESTAMP_STRUCT     c_since;
        char
c_credit[CREDIT_LEN+1];
        double
c_credit_lim;
        double
c_discount;
        double
c_balance;
        char
c_data[200+1];
} PAYMENT_DATA, *PPAYMENT_DATA;

typedef struct
{
        long
        ol_i_id;
        long
        ol_supply_w_id;
        short
        ol_quantity;
        double
        ol_amount;
        TIMESTAMP_STRUCT    ol_delivery_d;
} OL_ORDER_STATUS_DATA;

typedef struct
{       // input params
        long                w_id;
        short               d_id;
        long                c_id;
        char
c_last[LAST_NAME_LEN+1];

        // output params
        EXEC_STATUS
exec_status_code;
        char
c_first[FIRST_NAME_LEN+1];
        char
c_middle[MIDDLE_NAME_LEN+1];
        double              c_balance;
        long                o_id;
        TIMESTAMP_STRUCT    o_entry_d;
        short               o_carrier_id;
        OL_ORDER_STATUS_DATA
OL[MAX_OL_ORDER_STATUS_ITEMS];
        short               o_ol_cnt;
} ORDER_STATUS_DATA, *PORDER_STATUS_DATA;


typedef struct
{       // input params
        long                            w_id;
        short                           o_carrier_id;

        // output params
        EXEC_STATUS
        exec_status_code;
        SYSTEMTIME                      queue_time;
        long
        o_id[10];           // id's of delivered
orders for districts 1 to 10
} DELIVERY_DATA, *PDELIVERY_DATA;

//This structure is used for posting delivery
transactions and for writing them to the delivery
server.
typedef struct _DELIVERY_TRANSACTION
{
        SYSTEMTIME          queue;
        //time delivery transaction queued
        long                w_id;
        //delivery warehouse
        short               o_carrier_id;
        //carrier id
} DELIVERY_TRANSACTION;

typedef struct
{       // input params
        long
        w_id;
        short
        d_id;
```

```
          short
          threshold;

          // output params
          EXEC_STATUS
          exec_status_code;
          long
          low_stock;
} STOCK_LEVEL_DATA, *PSTOCK_LEVEL_DATA;
```

txn_base.h

```
/*        FILE:               TXN_BASE.H
 *                                    Microsoft
TPC-C Kit Ver. 4.69.000
 *                                    Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                                    Version
4.10.000 audited by Richard Gimarc, Performance
Metrics, 3/17/99
 *
 *        PURPOSE:  Header file for TPC-C txn class
implementation.
 *
 *  Change history:
 *                  4.20.000 - updated rev number to
match kit
 */

#pragma once

// need to declare functions for import, unless
define has already been created
// by the DLL's .cpp module for export.
#ifndef DllDecl
#define DllDecl  __declspec( dllimport )
#endif

class DllDecl CTPCC_BASE
{
          public:
                    CTPCC_BASE(void) {};
                    virtual ~CTPCC_BASE(void) {};

                    virtual PNEW_ORDER_DATA
          BuffAddr_NewOrder()           = 0;
                    virtual PPAYMENT_DATA
          BuffAddr_Payment()            = 0;
                    virtual PDELIVERY_DATA
          BuffAddr_Delivery()           = 0;
                    virtual PSTOCK_LEVEL_DATA
          BuffAddr_StockLevel()         = 0;
                    virtual PORDER_STATUS_DATA
          BuffAddr_OrderStatus()        = 0;

                    virtual void NewOrder
          () = 0;
                    virtual void Payment
          () = 0;
                    virtual void Delivery
          () = 0;
                    virtual void StockLevel
          () = 0;
                    virtual void OrderStatus      ()
= 0;
};
```

Appendix B:

Database Design

The TPC-C database was created with the following Transact-SQL scripts:

backup.sql

```
-----------------------------------------------------
-------------
--
--
--  File:   BACKUP.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.61
--
--          Copyright Microsoft, 2005
--
-----------------------------------------------------
-------------

DECLARE @startdate  DATETIME,
             @enddate        DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
                    CONVERT(VARCHAR(30),@startdate,
21)
DUMP DATABASE tpcc TO  tpccback1, tpccback3,
tpccback5, tpccback7, tpccback9, tpccback11,
tpccback13, tpccback15, tpccback17, tpccback19,
tpccback21, tpccback23, tpccback25, tpccback27,
tpccback29, tpccback31, tpccback33, tpccback35,
tpccback37, tpccback39, tpccback41, tpccback43,
tpccback45, tpccback47, tpccback49, tpccback51,
tpccback53  WITH init, stats = 1
SELECT  @enddate = GETDATE()
SELECT  'End date: ',
                    CONVERT(VARCHAR(30),@enddate, 21)
SELECT  'Elapsed time (in seconds): ',
                    DATEDIFF(second, @startdate,
@enddate)
GO
```

backupdev.sql

```
USE master
GO
EXEC sp_addumpdevice
'disk','tpccback1','c:\backup\1\tpccback1.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback3','c:\backup\3\tpccback3.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback5','c:\backup\5\tpccback5.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback7','c:\backup\7\tpccback7.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback9','c:\backup\9\tpccback9.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback11','c:\backup\11\tpccback11.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback13','c:\backup\13\tpccback13.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback15','c:\backup\15\tpccback15.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback17','c:\backup\17\tpccback17.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback19','c:\backup\19\tpccback19.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback21','c:\backup\21\tpccback21.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback23','c:\backup\23\tpccback23.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback25','c:\backup\25\tpccback25.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback27','c:\backup\27\tpccback27.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback29','c:\backup\29\tpccback29.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback31','c:\backup\31\tpccback31.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback33','c:\backup\33\tpccback33.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback35','c:\backup\35\tpccback35.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback37','c:\backup\37\tpccback37.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback39','c:\backup\39\tpccback39.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback41','c:\backup\41\tpccback41.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback43','c:\backup\43\tpccback43.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback45','c:\backup\45\tpccback45.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback47','c:\backup\47\tpccback47.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback49','c:\backup\49\tpccback49.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback51','c:\backup\51\tpccback51.dmp'
GO
EXEC sp_addumpdevice
'disk','tpccback53','c:\backup\53\tpccback53.dmp'
GO
```

createdb.sql

```
-----------------------------------------------------
-------------
--
--
--  File:   CREATEDB.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2005
--
--
--
-----------------------------------------------------
-------------

SET ANSI_NULL_DFLT_OFF ON
GO

USE master
GO

-----------------------------------
-- Create temporary table for timing
-----------------------------------
IF EXISTS( SELECT name FROM sysobjects WHERE name =
'tpcc_timer' )
    DROP TABLE tpcc_timer
GO

CREATE TABLE tpcc_timer
                (start_date   CHAR(30),
                 end_date     CHAR(30))
GO

INSERT INTO tpcc_timer VALUES(0,0)
GO

----------------------
-- Store starting time
----------------------
UPDATE  tpcc_timer
SET     start_date  = (SELECT   CONVERT(CHAR(30),
GETDATE(), 21))
GO


----------------------------
-- create main database files
----------------------------
CREATE DATABASE tpcc
ON PRIMARY
(         NAME               = MSSQL_tpcc_root,
          FILENAME  = 'c:\MSSQL_tpcc_root.mdf',
          SIZE               = 8MB,
```

```
          FILEGROWTH          = 0),


FILEGROUP MSSQL_fg
(         NAME                = MSSQL_1,
FILENAME  = 'c:\dev\tpcc_1\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_2,
FILENAME  = 'c:\dev\tpcc_2\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_3,
FILENAME  = 'c:\dev\tpcc_3\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_4,
FILENAME  = 'c:\dev\tpcc_4\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_5,
FILENAME  = 'c:\dev\tpcc_5\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_6,
FILENAME  = 'c:\dev\tpcc_6\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_7,
FILENAME  = 'c:\dev\tpcc_7\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_8,
FILENAME  = 'c:\dev\tpcc_8\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_9,
FILENAME  = 'c:\dev\tpcc_9\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_10,
FILENAME  = 'c:\dev\tpcc_10\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_11,
FILENAME  = 'c:\dev\tpcc_11\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_12,
FILENAME  = 'c:\dev\tpcc_12\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_13,
FILENAME  = 'c:\dev\tpcc_13\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_14,
FILENAME  = 'c:\dev\tpcc_14\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_15,
FILENAME  = 'c:\dev\tpcc_15\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_16,
FILENAME  = 'c:\dev\tpcc_16\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_17,
FILENAME  = 'c:\dev\tpcc_17\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_18,
FILENAME  = 'c:\dev\tpcc_18\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_19,
FILENAME  = 'c:\dev\tpcc_19\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_20,
FILENAME  = 'c:\dev\tpcc_20\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_21,
FILENAME  = 'c:\dev\tpcc_21\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_22,
FILENAME  = 'c:\dev\tpcc_22\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_23,
FILENAME  = 'c:\dev\tpcc_23\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_24,
FILENAME  = 'c:\dev\tpcc_24\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_25,
FILENAME  = 'c:\dev\tpcc_25\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_26,
FILENAME  = 'c:\dev\tpcc_26\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_27,
FILENAME  = 'c:\dev\tpcc_27\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_28,
FILENAME  = 'c:\dev\tpcc_28\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_29,
FILENAME  = 'c:\dev\tpcc_29\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_30,
FILENAME  = 'c:\dev\tpcc_30\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_31,
FILENAME  = 'c:\dev\tpcc_31\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_32,
FILENAME  = 'c:\dev\tpcc_32\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_33,
FILENAME  = 'c:\dev\tpcc_33\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_34,
FILENAME  = 'c:\dev\tpcc_34\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_35,
FILENAME  = 'c:\dev\tpcc_35\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_36,
FILENAME  = 'c:\dev\tpcc_36\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_37,
FILENAME  = 'c:\dev\tpcc_37\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_38,
FILENAME  = 'c:\dev\tpcc_38\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_39,
FILENAME  = 'c:\dev\tpcc_39\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_40,
FILENAME  = 'c:\dev\tpcc_40\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_41,
FILENAME  = 'c:\dev\tpcc_41\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_42,
FILENAME  = 'c:\dev\tpcc_42\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_43,
FILENAME  = 'c:\dev\tpcc_43\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_44,
FILENAME  = 'c:\dev\tpcc_44\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_45,
FILENAME  = 'c:\dev\tpcc_45\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_46,
FILENAME  = 'c:\dev\tpcc_46\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_47,
```

```
FILENAME  = 'c:\dev\tpcc_47\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_48,
FILENAME  = 'c:\dev\tpcc_48\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_49,
FILENAME  = 'c:\dev\tpcc_49\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_50,
FILENAME  = 'c:\dev\tpcc_50\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_51,
FILENAME  = 'c:\dev\tpcc_51\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_52,
FILENAME  = 'c:\dev\tpcc_52\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_53,
FILENAME  = 'c:\dev\tpcc_53\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_54,
FILENAME  = 'c:\dev\tpcc_54\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_55,
FILENAME  = 'c:\dev\tpcc_55\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_56,
FILENAME  = 'c:\dev\tpcc_56\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_57,
FILENAME  = 'c:\dev\tpcc_57\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_58,
FILENAME  = 'c:\dev\tpcc_58\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_59,
FILENAME  = 'c:\dev\tpcc_59\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_60,
FILENAME  = 'c:\dev\tpcc_60\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_61,
FILENAME  = 'c:\dev\tpcc_61\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_62,
FILENAME  = 'c:\dev\tpcc_62\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_63,
FILENAME  = 'c:\dev\tpcc_63\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_64,
FILENAME  = 'c:\dev\tpcc_64\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_65,
FILENAME  = 'c:\dev\tpcc_65\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_66,
FILENAME  = 'c:\dev\tpcc_66\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_67,
FILENAME  = 'c:\dev\tpcc_67\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_68,
FILENAME  = 'c:\dev\tpcc_68\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_69,
FILENAME  = 'c:\dev\tpcc_69\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_70,
FILENAME  = 'c:\dev\tpcc_70\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_71,
FILENAME  = 'c:\dev\tpcc_71\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_72,
FILENAME  = 'c:\dev\tpcc_72\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_73,
FILENAME  = 'c:\dev\tpcc_73\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_74,
FILENAME  = 'c:\dev\tpcc_74\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_75,
FILENAME  = 'c:\dev\tpcc_75\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_76,
FILENAME  = 'c:\dev\tpcc_76\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_77,
FILENAME  = 'c:\dev\tpcc_77\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_78,
FILENAME  = 'c:\dev\tpcc_78\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_79,
FILENAME  = 'c:\dev\tpcc_79\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_80,
FILENAME  = 'c:\dev\tpcc_80\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_81,
FILENAME  = 'c:\dev\tpcc_81\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_82,
FILENAME  = 'c:\dev\tpcc_82\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_83,
FILENAME  = 'c:\dev\tpcc_83\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_84,
FILENAME  = 'c:\dev\tpcc_84\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_85,
FILENAME  = 'c:\dev\tpcc_85\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_86,
FILENAME  = 'c:\dev\tpcc_86\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_87,
FILENAME  = 'c:\dev\tpcc_87\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_88,
FILENAME  = 'c:\dev\tpcc_88\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_89,
FILENAME  = 'c:\dev\tpcc_89\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_90,
FILENAME  = 'c:\dev\tpcc_90\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_91,
FILENAME  = 'c:\dev\tpcc_91\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_92,
FILENAME  = 'c:\dev\tpcc_92\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_93,
FILENAME  = 'c:\dev\tpcc_93\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_94,
FILENAME  = 'c:\dev\tpcc_94\',
```

```
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_95,
FILENAME  = 'c:\dev\tpcc_95\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_96,
FILENAME  = 'c:\dev\tpcc_96\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_97,
FILENAME  = 'c:\dev\tpcc_97\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_98,
FILENAME  = 'c:\dev\tpcc_98\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_99,
FILENAME  = 'c:\dev\tpcc_99\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_100,
FILENAME  = 'c:\dev\tpcc_100\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_101,
FILENAME  = 'c:\dev\tpcc_101\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_102,
FILENAME  = 'c:\dev\tpcc_102\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_103,
FILENAME  = 'c:\dev\tpcc_103\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_104,
FILENAME  = 'c:\dev\tpcc_104\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_105,
FILENAME  = 'c:\dev\tpcc_105\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_106,
FILENAME  = 'c:\dev\tpcc_106\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_107,
FILENAME  = 'c:\dev\tpcc_107\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_108,
FILENAME  = 'c:\dev\tpcc_108\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_109,
FILENAME  = 'c:\dev\tpcc_109\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_110,
FILENAME  = 'c:\dev\tpcc_110\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_111,
FILENAME  = 'c:\dev\tpcc_111\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_112,
FILENAME  = 'c:\dev\tpcc_112\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_113,
FILENAME  = 'c:\dev\tpcc_113\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_114,
FILENAME  = 'c:\dev\tpcc_114\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_115,
FILENAME  = 'c:\dev\tpcc_115\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_116,
FILENAME  = 'c:\dev\tpcc_116\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_117,
FILENAME  = 'c:\dev\tpcc_117\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_118,
FILENAME  = 'c:\dev\tpcc_118\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_119,
FILENAME  = 'c:\dev\tpcc_119\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_120,
FILENAME  = 'c:\dev\tpcc_120\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_121,
FILENAME  = 'c:\dev\tpcc_121\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_122,
FILENAME  = 'c:\dev\tpcc_122\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_123,
FILENAME  = 'c:\dev\tpcc_123\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_124,
FILENAME  = 'c:\dev\tpcc_124\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_125,
FILENAME  = 'c:\dev\tpcc_125\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_126,
FILENAME  = 'c:\dev\tpcc_126\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_127,
FILENAME  = 'c:\dev\tpcc_127\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_128,
FILENAME  = 'c:\dev\tpcc_128\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_129,
FILENAME  = 'c:\dev\tpcc_129\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_130,
FILENAME  = 'c:\dev\tpcc_130\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_131,
FILENAME  = 'c:\dev\tpcc_131\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_132,
FILENAME  = 'c:\dev\tpcc_132\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_133,
FILENAME  = 'c:\dev\tpcc_133\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_134,
FILENAME  = 'c:\dev\tpcc_134\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_135,
FILENAME  = 'c:\dev\tpcc_135\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_136,
FILENAME  = 'c:\dev\tpcc_136\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_137,
FILENAME  = 'c:\dev\tpcc_137\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_138,
FILENAME  = 'c:\dev\tpcc_138\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_139,
FILENAME  = 'c:\dev\tpcc_139\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_140,
FILENAME  = 'c:\dev\tpcc_140\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_141,
FILENAME  = 'c:\dev\tpcc_141\',
          SIZE                = 70990MB,
```

```
          FILEGROWTH          = 0),
(         NAME                = MSSQL_142,
FILENAME  = 'c:\dev\tpcc_142\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_143,
FILENAME  = 'c:\dev\tpcc_143\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_144,
FILENAME  = 'c:\dev\tpcc_144\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_145,
FILENAME  = 'c:\dev\tpcc_145\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_146,
FILENAME  = 'c:\dev\tpcc_146\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_147,
FILENAME  = 'c:\dev\tpcc_147\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_148,
FILENAME  = 'c:\dev\tpcc_148\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_149,
FILENAME  = 'c:\dev\tpcc_149\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_150,
FILENAME  = 'c:\dev\tpcc_150\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_151,
FILENAME  = 'c:\dev\tpcc_151\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_152,
FILENAME  = 'c:\dev\tpcc_152\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_153,
FILENAME  = 'c:\dev\tpcc_153\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_154,
FILENAME  = 'c:\dev\tpcc_154\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_155,
FILENAME  = 'c:\dev\tpcc_155\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_156,
FILENAME  = 'c:\dev\tpcc_156\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_157,
FILENAME  = 'c:\dev\tpcc_157\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_158,
FILENAME  = 'c:\dev\tpcc_158\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_159,
FILENAME  = 'c:\dev\tpcc_159\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_160,
FILENAME  = 'c:\dev\tpcc_160\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_161,
FILENAME  = 'c:\dev\tpcc_161\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0),
(         NAME                = MSSQL_162,
FILENAME  = 'c:\dev\tpcc_162\',
          SIZE                = 70990MB,
          FILEGROWTH          = 0)


LOG ON
(         NAME                = MSSQL_tpcc_log_1,
          FILENAME  = 'E:',
          SIZE                = 500000MB,
          FILEGROWTH          = 0)

COLLATE Latin1_General_BIN
GO

--------------------
-- Store ending time
--------------------
UPDATE  tpcc_timer
SET     end_date    = (SELECT   CONVERT(CHAR(30),
GETDATE(), 21))
GO

SELECT  DATEDIFF(second,(SELECT start_date FROM
tpcc_timer),(SELECT end_date FROM tpcc_timer))
GO

------------------------
-- remove temporary table
------------------------
IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_timer' )
    DROP TABLE tpcc_timer
GO
```

dbopt1.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   DBOPT1.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Sets database options for load
--
--
--
-------------------------------------------------------
-------------
USE master
GO

ALTER DATABASE tpcc SET RECOVERY BULK_LOGGED
GO

EXEC sp_dboption tpcc,'trunc. log on chkpt.',TRUE
GO

ALTER DATABASE tpcc SET TORN_PAGE_DETECTION OFF
GO

ALTER DATABASE tpcc SET PAGE_VERIFY NONE
GO

USE tpcc
GO

CHECKPOINT
GO
```

dbopt2.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   DBOPT2.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Sets database options after load
--
--
--
-------------------------------------------------------
-------------
ALTER DATABASE tpcc SET RECOVERY FULL
GO

USE tpcc
GO
```

```
CHECKPOINT
GO

sp_configure 'allow updates',1
GO

RECONFIGURE WITH OVERRIDE
GO

DECLARE   @msg                varchar(50)

-----------------------------------------------
--        OPTIONS FOR SQL SERVER 2000         --
-- Set option values for user-defined indexes --
-----------------------------------------------

SET     @msg    = ' '
PRINT   @msg
SET     @msg    = 'Setting SQL Server indexoptions'
PRINT   @msg
SET     @msg    = ' '
PRINT   @msg

EXEC sp_indexoption 'customer',
'DisAllowPageLocks',          TRUE
EXEC sp_indexoption 'district',
          'DisAllowPageLocks',          TRUE
EXEC sp_indexoption 'warehouse',
'DisAllowPageLocks',          TRUE
EXEC sp_indexoption 'stock',
'DisAllowPageLocks',          TRUE
EXEC sp_indexoption 'order_line',
'DisAllowRowLocks',     TRUE
EXEC sp_indexoption 'orders',
'DisAllowRowLocks',     TRUE
EXEC sp_indexoption 'new_order',
'DisAllowRowLocks',     TRUE
EXEC sp_indexoption 'item',
'DisAllowRowLocks',     TRUE
EXEC sp_indexoption 'item',
'DisAllowPageLocks',    False
GO

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

SELECT  name,
        lockflags
FROM    sysindexes
WHERE   object_id('warehouse')  = id OR
        object_id('district')   = id OR
        object_id('customer')   = id OR
        object_id('stock')      = id OR
        object_id('orders')     = id OR
        object_id('order_line') = id OR
        object_id('history')    = id OR
        object_id('new_order')  = id OR
        object_id('item')       = id
ORDER   BY lockflags asc
GO

sp_configure 'allow updates',0
GO

RECONFIGURE WITH OVERRIDE
GO

EXEC sp_dboption tpcc,        'auto update
statistics',        FALSE
EXEC sp_dboption tpcc,        'auto create
statistics',        FALSE
GO

DECLARE @db_id  int,
        @tbl_id int

SET     @db_id  = DB_ID('tpcc')
SET     @tbl_id = OBJECT_ID('tpcc..warehouse')
DBCC PINTABLE (@db_id, @tbl_id)

SET     @tbl_id = OBJECT_ID('tpcc..district')
DBCC PINTABLE (@db_id, @tbl_id)

SET     @tbl_id = OBJECT_ID('tpcc..new_order')
DBCC PINTABLE (@db_id, @tbl_id)

SET     @tbl_id = OBJECT_ID('tpcc..item')
DBCC PINTABLE (@db_id, @tbl_id)
GO
```

delivery.sql

```
-----------------------------------------------------
-------------
--
--
--  File:   DELIVERY.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates delivery stored procedure
--
--
--
--          Interface Level:    4.20.000
--
--
--
-----------------------------------------------------
-------------
SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

USE tpcc
GO

IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_delivery' )
    DROP PROCEDURE tpcc_delivery
GO

CREATE PROC tpcc_delivery
                @w_id           int,

                @o_carrier_id   smallint

AS

DECLARE @d_id       tinyint,
        @o_id       int,
        @c_id       int,
        @total      money,
        @oid1       int,
        @oid2       int,
        @oid3       int,
        @oid4       int,
        @oid5       int,
        @oid6       int,
        @oid7       int,
        @oid8       int,
        @oid9       int,
        @oid10      int

SELECT  @d_id   = 0

BEGIN TRANSACTION d
    WHILE (@d_id < 10)
    BEGIN
        SELECT  @d_id   = @d_id + 1,
                @total  = 0,
                @o_id   = 0

        SELECT  TOP 1
                @o_id   = no_o_id
        FROM    new_order WITH  (serializable
updlock)
        WHERE   no_w_id = @w_id AND
                no_d_id = @d_id
        ORDER   BY no_o_id ASC

        IF (@@rowcount <> 0)
        BEGIN
            --  claim the order for this district
            DELETE  new_order
            WHERE   no_w_id = @w_id AND
                    no_d_id = @d_id AND
                    no_o_id = @o_id
```

```
            --  set carrier_id on this order (and get
customer id)
            UPDATE  orders
            SET     o_carrier_id    = @o_carrier_id,
                    @c_id           = o_c_id
            WHERE   o_w_id          = @w_id AND
                    o_d_id          = @d_id AND
                    o_id            = @o_id

            --  set date in all lineitems for this
order (and sum amounts)
            UPDATE  order_line
            SET     ol_delivery_d   = GETDATE(),
                    @total          = @total +
ol_amount
            WHERE   ol_w_id         = @w_id AND
                    ol_d_id         = @d_id AND
                    ol_o_id         = @o_id

            --  accummulate lineitem amounts for this
order into customer
            UPDATE  customer
            SET     c_balance       = c_balance +
@total,
                    c_delivery_cnt  = c_delivery_cnt
+ 1
            WHERE   c_w_id          = @w_id AND
                    c_d_id          = @d_id AND
                    c_id            = @c_id
        END

    SELECT  @oid1   = CASE @d_id WHEN   1   THEN
@o_id ELSE @oid1   END,
            @oid2   = CASE @d_id WHEN   2   THEN
@o_id ELSE @oid2   END,
            @oid3   = CASE @d_id WHEN   3   THEN
@o_id ELSE @oid3   END,
            @oid4   = CASE @d_id WHEN   4   THEN
@o_id ELSE @oid4   END,
            @oid5   = CASE @d_id WHEN   5   THEN
@o_id ELSE @oid5   END,
            @oid6   = CASE @d_id WHEN   6   THEN
@o_id ELSE @oid6   END,
            @oid7   = CASE @d_id WHEN   7   THEN
@o_id ELSE @oid7   END,
            @oid8   = CASE @d_id WHEN   8   THEN
@o_id ELSE @oid8   END,
            @oid9   = CASE @d_id WHEN   9   THEN
@o_id ELSE @oid9   END,
            @oid10  = CASE @d_id WHEN  10   THEN
@o_id ELSE @oid10  END
    END

COMMIT TRANSACTION d

-- return delivery data to client

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

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO
```

getargs.c

```
//        File:               GETARGS.C
//                                      Microsoft
TPC-C Kit Ver. 4.51
//                                      Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000, 2001, 2002,
2003
//        Purpose:  Source file for command line
processing


// Includes
#include "tpcc.h"

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

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

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

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

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

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

                    ptr = argv[i];

                    switch (ptr[1])
                    {
                    case '?':     /* Fall throught */

            GetArgsLoaderUsage();
                                            break;

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

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

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

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

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

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

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

                    case 't':
                                        {

          pargs->tables_all = FALSE;
                                                  if
(strcmp(ptr+2,"item") == 0)

          pargs->table_item = TRUE;

          else if (strcmp(ptr+2,"warehouse") == 0)

          pargs->table_warehouse = TRUE;

          else if (strcmp(ptr+2,"customer") == 0)

          pargs->table_customer = TRUE;

          else if (strcmp(ptr+2,"orders") == 0)

          pargs->table_orders = TRUE;

          else
                                                  {

          printf("\nUnrecognized command");

          GetArgsLoaderUsage();

          exit(1);
                                                  }


          break;
                                        }

                              case 'f':
                                        pargs-
>loader_res_file = ptr+2;
                                        break;

                              case 'L':
                                        pargs-
>log_path = ptr+2;
                                        break;

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

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

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

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

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

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

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

    return;
}


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

void GetArgsLoaderUsage()
{

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


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

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

    exit(0);
}
```

idxcuscl.sql

```
-----------------------------------------------------
-------------
--
--
```

```
--  File:   IDXCUSCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on customer table
--
-------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'customer_c1' )
    DROP INDEX customer.customer_c1

CREATE UNIQUE CLUSTERED INDEX customer_c1 ON
customer(c_w_id, c_d_id, c_id)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxcusnc.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   IDXCUSNC.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates non-clustered index on customer
table       --
-------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'customer_nc1' )
    DROP INDEX customer.customer_nc1

CREATE UNIQUE NONCLUSTERED INDEX customer_nc1 ON
customer(c_w_id, c_d_id, c_last, c_first, c_id)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxdiscl.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   IDXDISCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on district table
--
-------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'district_c1' )
    DROP INDEX district.district_c1

CREATE UNIQUE CLUSTERED INDEX district_c1 ON
district(d_w_id, d_id)
    WITH FILLFACTOR=100 ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxhiscl.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   IDXHISCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on history table
--
--
--
--          CAUTION: This index is only beneficial
for systems  --
--          CAUTION: with 8 or more processors.
--
--          CAUTION: It may negatively impact
performance on    --
--          CAUTION: systems with less than 8
processors.       --
--
--
-------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'history_c1' )
    DROP INDEX history.history_c1

CREATE UNIQUE CLUSTERED INDEX history_c1 ON
history(h_c_w_id, h_date, h_c_d_id, h_c_id, h_amount)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxitmcl.sql

```
------------------------------------------------------
-------------
--
--
--  File:   IDXITMCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on item table
--
--
--
------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'item_c1' )
    DROP INDEX item.item_c1

CREATE UNIQUE CLUSTERED INDEX item_c1 ON item(i_id)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxnodcl.sql

```
------------------------------------------------------
-------------
--
--
--  File:   IDXNODCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on new-order
table          --
--
--
------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'new_order_c1' )
    DROP INDEX new_order.new_order_c1

CREATE UNIQUE CLUSTERED INDEX new_order_c1 ON
new_order(no_w_id, no_d_id, no_o_id)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxodlcl.sql

```
------------------------------------------------------
-------------
--
--
--  File:   IDXODLCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on order-line
table         --
--
--
------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'order_line_c1' )
    DROP INDEX order_line.order_line_c1

CREATE UNIQUE CLUSTERED INDEX order_line_c1 ON
order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxordcl.sql

```
------------------------------------------------------
-------------
--
--
--  File:   IDXORDCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on orders table
--
--
--
------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'orders_c1' )
    DROP INDEX orders.orders_c1

CREATE UNIQUE CLUSTERED INDEX orders_c1 ON
orders(o_w_id, o_d_id, o_id)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxordnc.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   IDXORDNC.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates non-clustered index on orders
table         --
-------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'orders_nc1' )
    DROP INDEX orders.orders_nc1

CREATE INDEX orders_nc1 ON orders(o_w_id, o_d_id,
o_c_id, o_id)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxstkcl.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   IDXSTKCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on stock table
--
--
-------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'stock_c1' )
    DROP INDEX stock.stock_c1

CREATE UNIQUE CLUSTERED INDEX stock_c1 ON
stock(s_i_id, s_w_id)
    ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

idxwarcl.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   IDXWARCL.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates clustered index on warehouse
table          --
--
--
-------------------------------------------------------
-------------
USE tpcc
GO

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
        CONVERT(VARCHAR(30),@startdate,21)

IF EXISTS ( SELECT name FROM sysindexes WHERE name =
'warehouse_c1' )
    DROP INDEX warehouse.warehouse_c1

CREATE UNIQUE CLUSTERED INDEX warehouse_c1 ON
warehouse(w_id)
    WITH FILLFACTOR=100 ON MSSQL_fg

SELECT  @enddate = GETDATE()
SELECT  'End date:',
        CONVERT(VARCHAR(30),@enddate,21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

NewOrd.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   NEWORD.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates neworder stored procedure
--
--
--
--          Interface Level:    4.20.000
--
--
--
-------------------------------------------------------
-------------
SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

USE tpcc
GO

IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_neworder' )
    DROP PROCEDURE tpcc_neworder
GO

CREATE PROCEDURE    tpcc_neworder
                        @w_id           int,
                        @d_id           tinyint,
                        @c_id           int,
                        @o_ol_cnt       tinyint,
                        @o_all_local    tinyint,
```

```
                    @i_id1  int = 0, @s_w_id1
int = 0, @ol_qty1  smallint = 0,
                    @i_id2  int = 0, @s_w_id2
int = 0, @ol_qty2  smallint = 0,
                    @i_id3  int = 0, @s_w_id3
int = 0, @ol_qty3  smallint = 0,
                    @i_id4  int = 0, @s_w_id4
int = 0, @ol_qty4  smallint = 0,
                    @i_id5  int = 0, @s_w_id5
int = 0, @ol_qty5  smallint = 0,
                    @i_id6  int = 0, @s_w_id6
int = 0, @ol_qty6  smallint = 0,
                    @i_id7  int = 0, @s_w_id7
int = 0, @ol_qty7  smallint = 0,
                    @i_id8  int = 0, @s_w_id8
int = 0, @ol_qty8  smallint = 0,
                    @i_id9  int = 0, @s_w_id9
int = 0, @ol_qty9  smallint = 0,
                    @i_id10 int = 0, @s_w_id10
int = 0, @ol_qty10 smallint = 0,
                    @i_id11 int = 0, @s_w_id11
int = 0, @ol_qty11 smallint = 0,
                    @i_id12 int = 0, @s_w_id12
int = 0, @ol_qty12 smallint = 0,
                    @i_id13 int = 0, @s_w_id13
int = 0, @ol_qty13 smallint = 0,
                    @i_id14 int = 0, @s_w_id14
int = 0, @ol_qty14 smallint = 0,
                    @i_id15 int = 0, @s_w_id15
int = 0, @ol_qty15 smallint = 0



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

BEGIN

BEGIN TRANSACTION n

------------------------------------------------------
------
```

```
--  get district tax and next availible order id and
update
--  plus initialize local variables
------------------------------------------------------
------
    UPDATE  district
    SET     @d_tax          = d_tax,
            @o_id           = d_next_o_id,
            d_next_o_id     = d_next_o_id + 1,
            @o_entry_d      = GETDATE(),
            @li_no          = 0,
            @commit_flag    = 1
    WHERE   d_w_id          = @w_id AND
            d_id            = @d_id

----------------------
--  process orderlines
----------------------
    WHILE (@li_no < @o_ol_cnt)
    BEGIN
        SELECT  @li_no  = @li_no + 1

-----------------------------------------------
--  set i_id, s_w_id, and qty for this lineitem
-----------------------------------------------
        SELECT  @li_id = CASE @li_no
                                WHEN 1 THEN @i_id1
                                WHEN 2 THEN @i_id2
                                WHEN 3 THEN @i_id3
                                WHEN 4 THEN @i_id4
                                WHEN 5 THEN @i_id5
                                WHEN 6 THEN @i_id6
                                WHEN 7 THEN @i_id7
                                WHEN 8 THEN @i_id8
                                WHEN 9 THEN @i_id9
                                WHEN 10 THEN @i_id10
                                WHEN 11 THEN @i_id11
                                WHEN 12 THEN @i_id12
                                WHEN 13 THEN @i_id13
                                WHEN 14 THEN @i_id14
                                WHEN 15 THEN @i_id15
                        END,

                @li_s_w_id = CASE @li_no
                                WHEN 1 THEN @s_w_id1
                                WHEN 2 THEN @s_w_id2
                                WHEN 3 THEN @s_w_id3
                                WHEN 4 THEN @s_w_id4
                                WHEN 5 THEN @s_w_id5
                                WHEN 6 THEN @s_w_id6
                                WHEN 7 THEN @s_w_id7
                                WHEN 8 THEN @s_w_id8
                                WHEN 9 THEN @s_w_id9
                                WHEN 10 THEN
@s_w_id10
                                WHEN 11 THEN
@s_w_id11
                                WHEN 12 THEN
@s_w_id12
                                WHEN 13 THEN
@s_w_id13
                                WHEN 14 THEN
@s_w_id14
```

```
                                WHEN 15 THEN
@s_w_id15
                        END,

                @li_qty = CASE  @li_no
                                WHEN 1 THEN @ol_qty1
                                WHEN 2 THEN @ol_qty2
                                WHEN 3 THEN @ol_qty3
                                WHEN 4 THEN @ol_qty4
                                WHEN 5 THEN @ol_qty5
                                WHEN 6 THEN @ol_qty6
                                WHEN 7 THEN @ol_qty7
                                WHEN 8 THEN @ol_qty8
                                WHEN 9 THEN @ol_qty9
                                WHEN 10 THEN
@ol_qty10
                                WHEN 11 THEN
@ol_qty11
                                WHEN 12 THEN
@ol_qty12
                                WHEN 13 THEN
@ol_qty13
                                WHEN 14 THEN
@ol_qty14
                                WHEN 15 THEN
@ol_qty15
                        END

--------------------------------------
--  get item data (no one updates item)
--------------------------------------
        SELECT  @i_price    = i_price,
                @i_name     = i_name,
                @i_data     = i_data
        FROM    item WITH (repeatableread)
        WHERE   i_id        = @li_id

----------------------
--  update stock values
----------------------
        UPDATE  stock
        SET     s_ytd           = s_ytd + @li_qty,
                @s_quantity     = s_quantity   =
s_quantity - @li_qty +
                                  CASE WHEN
(s_quantity - @li_qty < 10) THEN 91 ELSE 0 END,
                s_order_cnt     = s_order_cnt + 1,
                s_remote_cnt    = s_remote_cnt +
                                  CASE WHEN
(@li_s_w_id = @w_id) THEN 0 ELSE 1 END,
                @s_data         = s_data,
                @s_dist         = CASE @d_id
                                    WHEN 1  THEN
s_dist_01
                                    WHEN 2  THEN
s_dist_02
                                    WHEN 3  THEN
s_dist_03
                                    WHEN 4  THEN
s_dist_04
                                    WHEN 5  THEN
s_dist_05
```

```
                                WHEN 6  THEN
s_dist_06
                                WHEN 7  THEN
s_dist_07
                                WHEN 8  THEN
s_dist_08
                                WHEN 9  THEN
s_dist_09
                                WHEN 10 THEN
s_dist_10
                              END
        WHERE   s_i_id          = @li_id AND
                s_w_id          = @li_s_w_id

-------------------------------------------------------
-----------------
--  if there actually is a stock (and item) with
these ids, go to work
-------------------------------------------------------
----------------
        IF (@@rowcount > 0)
        BEGIN
-------------------------------------------------------
------
--  insert order_line data (using data from item and
stock)
-------------------------------------------------------
------
            INSERT INTO order_line VALUES(  @o_id,
                                            @d_id,

                                            @w_id,
                                            @li_no,
                                            @li_id,
                                            'dec 31,
1899',
                                            @i_price
* @li_qty,

@li_s_w_id,
                                            @li_qty,
                                            @s_dist)

--------------------------------
--  send line-item data to client
--------------------------------
            SELECT  @i_name,
                    @s_quantity,
                    b_g = CASE  WHEN (
(patindex('%ORIGINAL%',@i_data) > 0) AND


(patindex('%ORIGINAL%',@s_data) > 0) )

THEN 'B' ELSE 'G' END,
                    @i_price,
                    @i_price * @li_qty
                    END
                    ELSE
                    BEGIN
-------------------------------------------------------
-----
--  no item (or stock) found - triggers rollback
condition
-------------------------------------------------------
-----
            SELECT  '',0,'',0,0
            SELECT  @commit_flag    = 0
                    END
          END
-------------------------------------------------------
--
--  get customer last name, discount, and credit
rating
-------------------------------------------------------
--
    SELECT  @c_last     = c_last,
            @c_discount = c_discount,
            @c_credit   = c_credit,
            @c_id_local = c_id
    FROM    customer WITH (repeatableread)
    WHERE   c_id        = @c_id AND
            c_w_id      = @w_id AND
            c_d_id      = @d_id


--------------------------------------
--  insert fresh row into orders table
--------------------------------------
    INSERT INTO orders VALUES ( @o_id,
                                @d_id,
                                @w_id,
                                @c_id_local,
                                0,
                                @o_ol_cnt,
                                @o_all_local,
                                @o_entry_d)


-------------------------------------------------
--  insert corresponding row into new-order table
-------------------------------------------------
    INSERT INTO new_order VALUES (  @o_id,
                                    @d_id,
                                    @w_id)


------------------------
--  select warehouse tax
------------------------
    SELECT  @w_tax  = w_tax
    FROM    warehouse WITH (repeatableread)
    WHERE   w_id    = @w_id

    IF (@commit_flag = 1)

                    COMMIT TRANSACTION n
    ELSE
-------------------------------
--  all that work for nuthin!!!
-------------------------------
        ROLLBACK TRANSACTION n

-------------------------------
--  return order data to client
-------------------------------
    SELECT  @w_tax,
            @d_tax,
            @o_id,
            @c_last,
            @c_discount,
            @c_credit,
            @o_entry_d,
            @commit_flag
END
GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO
```

null-txns.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   NULL-TXNS.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          This script will create stored procs
which         --
--          accept the same parameters and return
correctly     --
--          formed results sets to match the standard
TPC-C     --
--          stored procs.  Of course, the advantage
is that     --
--          these stored procs place almost no load
on          --
--          SQL Server and do not require a database.
--
--
--
--          Interface Level:    4.10.000
--
--
--
-------------------------------------------------------
-------------
USE tpcc
GO

IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_delivery' )
    DROP PROCEDURE tpcc_delivery
GO
IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_neworder' )
    DROP PROCEDURE tpcc_neworder
GO
```

```
IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_orderstatus' )
    DROP PROCEDURE tpcc_orderstatus
GO
IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_payment' )
    DROP PROCEDURE tpcc_payment
GO
IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_stocklevel' )
    DROP PROCEDURE tpcc_stocklevel
GO
IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_version' )
    DROP PROCEDURE tpcc_version
GO
IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'order_line_null' )
    DROP PROCEDURE order_line_null
GO

CREATE PROCEDURE    tpcc_delivery
                        @w_id           int,

                        @o_carrier_id   smallint

AS

DECLARE @d_id       tinyint,
        @o_id       int,
        @c_id       int,
        @total      numeric(12,2),
        @oid1       int,
        @oid2       int,
        @oid3       int,
        @oid4       int,
        @oid5       int,
        @oid6       int,
        @oid7       int,
        @oid8       int,
        @oid9       int,
        @oid10      int,
        @delaytime  varchar(30)

-----------------------------------------------------
-- uniform random delay of 0 - 1 second; avg = 0.50
-----------------------------------------------------
SELECT  @delaytime = '00:00:0' +
CAST(CAST((RAND()*1.00) AS decimal(4,3)) AS char(5))

WAITFOR delay @delaytime

SELECT  3001, 3001, 3001, 3001, 3001, 3001, 3001,
3001, 3001, 3001
GO

CREATE PROCEDURE    tpcc_neworder
                        @w_id           int,
                        @d_id           tinyint,
                        @c_id           int,
                        @o_ol_cnt       tinyint,
                        @o_all_local    tinyint,
                        @i_id1  int = 0, @s_w_id1 int
= 0, @ol_qty1 smallint = 0,
                        @i_id2  int = 0, @s_w_id2 int
= 0, @ol_qty2 smallint = 0,
                        @i_id3  int = 0, @s_w_id3 int
= 0, @ol_qty3 smallint = 0,
                        @i_id4  int = 0, @s_w_id4 int
= 0, @ol_qty4 smallint = 0,
                        @i_id5  int = 0, @s_w_id5 int
= 0, @ol_qty5 smallint = 0,
                        @i_id6  int = 0, @s_w_id6 int
= 0, @ol_qty6 smallint = 0,
                        @i_id7  int = 0, @s_w_id7 int
= 0, @ol_qty7 smallint = 0,
                        @i_id8  int = 0, @s_w_id8 int
= 0, @ol_qty8 smallint = 0,
                        @i_id9  int = 0, @s_w_id9 int
= 0, @ol_qty9 smallint = 0,
                        @i_id10 int = 0, @s_w_id10
int = 0, @ol_qty10 smallint = 0,
                        @i_id11 int = 0, @s_w_id11
int = 0, @ol_qty11 smallint = 0,
                        @i_id12 int = 0, @s_w_id12
int = 0, @ol_qty12 smallint = 0,
                        @i_id13 int = 0, @s_w_id13
int = 0, @ol_qty13 smallint = 0,
                        @i_id14 int = 0, @s_w_id14
int = 0, @ol_qty14 smallint = 0,
                        @i_id15 int = 0, @s_w_id15
int = 0, @ol_qty15 smallint = 0



AS
DECLARE @w_tax          numeric(4,4),
        @d_tax          numeric(4,4),
        @c_last         char(16),
        @c_credit       char(2),
        @c_discount     numeric(4,4),
        @i_price        numeric(5,2),
        @i_name         char(24),
        @o_entry_d      datetime,
        @li_no          int,
        @o_id           int,
        @commit_flag    tinyint,
        @li_id          int,
        @li_qty         smallint,
        @delaytime      varchar(30)

BEGIN
    ------------------------------------------------
---
    -- uniform random delay of 0 - 0.6 second; avg =
0.3
    ------------------------------------------------
---
    SELECT  @delaytime  = '00:00:0' +
CAST(CAST((RAND()*0.60) AS decimal(4,3)) AS char(5))

    WAITFOR delay @delaytime

    ----------------------
    --  process orderlines
    ----------------------
SELECT  @commit_flag    = 1,
        @li_no          = 0

WHILE (@li_no < @o_ol_cnt)
BEGIN
    SELECT  @li_id  = CASE @li_no
                          WHEN  1 THEN @i_id1
                          WHEN  2 THEN @i_id2
                          WHEN  3 THEN @i_id3
                          WHEN  4 THEN @i_id4
                          WHEN  5 THEN @i_id5
                          WHEN  6 THEN @i_id6
                          WHEN  7 THEN @i_id7
                          WHEN  8 THEN @i_id8
                          WHEN  9 THEN @i_id9
                          WHEN 10 THEN @i_id10
                          WHEN 11 THEN @i_id11
                          WHEN 12 THEN @i_id12
                          WHEN 13 THEN @i_id13
                          WHEN 14 THEN @i_id14
                          WHEN 15 THEN @i_id15
                      END

    SELECT  @li_no      = @li_no + 1


    SELECT  @i_price    = 23.45, @li_qty = @li_no

    IF (@li_id = 999999)
    BEGIN
        SELECT  '',0,'',0,0

        SELECT  @commit_flag = 0
    END
    ELSE
    BEGIN
        SELECT  'Item Name blah',
                17,
                'G',
                @i_price,
                @i_price * @li_qty
    END
END

------------------------------
--  return order data to client
------------------------------
SELECT  @w_tax      = 0.1234,
        @d_tax      = 0.0987,
        @o_id       = 3001,
        @c_last     = 'BAROUGHTABLE',
        @c_discount = 0.2198,
        @c_credit   = 'GC',
        @o_entry_d  = GETDATE()

SELECT  @w_tax,
        @d_tax,
        @o_id,
        @c_last,
        @c_discount,
        @c_credit,
        @o_entry_d,
```

```
            @commit_flag

END
GO

CREATE PROCEDURE    tpcc_orderstatus
                        @w_id       int,
                        @d_id       tinyint,


                        @c_id       int,
                        @c_last     char(16) = ''


AS
DECLARE @c_balance      numeric(12,2),
        @c_first        char(16),
        @c_middle       char(2),
        @o_id           int,
        @o_entry_d      datetime,
        @o_carrier_id   smallint,
        @ol_cnt         smallint,
        @delaytime      varchar(30)

-----------------------------------------------------
-- uniform random delay of 0 - 0.2 second; avg = 0.1
-----------------------------------------------------
SELECT  @delaytime  = '00:00:0' +
CAST(CAST((RAND()*0.20) AS decimal(4,3)) AS char(5))

WAITFOR delay @delaytime

SELECT  @c_id       = 113,
        @c_balance  = -10.00,
        @c_first    = '8YCodgytqCj8',
        @c_middle   = 'OE',
        @c_last     = 'OUGHTOUGHTABLE',
        @o_id       = 3456,
        @o_entry_d  = GETDATE(),
        @o_carrier_id   = 1

SELECT  @ol_cnt = (RAND() * 11) + 5

SET     ROWCOUNT @ol_cnt

SELECT  ol_supply_w_id,
        ol_i_id,
        ol_quantity,
        ol_amount,
        ol_delivery_d
FROM    order_line_null

SELECT  @c_id,
        @c_last,
        @c_first,
        @c_middle,
        @o_entry_d,
        @o_carrier_id,
        @c_balance,
        @o_id
GO

CREATE PROCEDURE    tpcc_payment
                        @w_id       int,
                        @c_w_id     int,
                        @h_amount   numeric(6,2),
                        @d_id       tinyint,
                        @c_d_id     tinyint,
                        @c_id       int,
                        @c_last     char(16) = ''


AS
DECLARE @w_street_1     char(20),
        @w_street_2     char(20),
        @w_city         char(20),
        @w_state        char(2),
        @w_zip          char(9),
        @w_name         char(10),
        @d_street_1     char(20),
        @d_street_2     char(20),
        @d_city         char(20),
        @d_state        char(2),
        @d_zip          char(9),
        @d_name         char(10),
        @c_first        char(16),
        @c_middle       char(2),
        @c_street_1     char(20),
        @c_street_2     char(20),
        @c_city         char(20),
        @c_state        char(2),
        @c_zip          char(9),
        @c_phone        char(16),
        @c_since        datetime,
        @c_credit       char(2),
        @c_credit_lim   numeric(12,2),
        @c_balance      numeric(12,2),
        @c_discount     numeric(4,4),
        @data           char(500),
        @c_data         char(500),
        @datetime       datetime,
        @w_ytd          numeric(12,2),
        @d_ytd          numeric(12,2),
        @cnt            smallint,
        @val            smallint,
        @screen_data          char(200),
        @d_id_local     tinyint,
        @w_id_local     int,
        @c_id_local     int,
        @delaytime      varchar(30)

-----------------------------------------------------
-- uniform random delay of 0 - 0.3 second; avg = 0.15
-----------------------------------------------------
SELECT  @delaytime  = '00:00:0' +
CAST(CAST((RAND()*0.20) AS decimal(4,3)) AS char(5))

WAITFOR delay @delaytime

SELECT  @screen_data    = ''

-----------------------------------------
--  get customer info and update balances
-----------------------------------------
SELECT  @d_street_1 = 'rqSHHakqyV',
        @d_street_2 = 'zZ98nW3BR2s',
        @d_city     = 'ArNr4GNFV9',
        @d_state    = 'aV',
        @d_zip      = '453511111'

-----------------------------------------------
--  get warehouse data and update year-to-date
-----------------------------------------------
SELECT  @w_street_1 = 'rqSHHakqyV',
        @w_street_2 = 'zZ98nW3BR2s',
        @w_city     = 'ArNr4GNFV9',
        @w_state    = 'aV',
        @w_zip      = '453511111'

SELECT  @c_id              = 123,
        @c_balance     = -10000.00,
        @c_first       = 'KmR03Xureb',
        @c_middle      = 'OE',
        @c_last        = 'BAROUGHTBAR',
        @c_street_1    = 'QpGdOHjv8mR9vNI8V',
        @c_street_2    = 'dzKoCObBqbC3yu',
        @c_city        = 'zAKZXdC037FQxq',
        @c_state       = 'QA',
        @c_zip         = '700311111',
        @c_phone       = '2967264064528555',
        @c_credit      = 'GC',
        @c_credit_lim  = 50000.00,
        @c_discount    = 0.3069,
        @c_since       = GETDATE(),
        @datetime      = GETDATE()

------------------------
--  return data to client
------------------------
SELECT  @c_id,
        @c_last,
        @datetime,
        @w_street_1,
        @w_street_2,
        @w_city,
        @w_state,
        @w_zip,
        @d_street_1,
        @d_street_2,
        @d_city,
        @d_state,
        @d_zip,
        @c_first,
        @c_middle,
        @c_street_1,
        @c_street_2,
        @c_city,
        @c_state,
        @c_zip,
        @c_phone,
        @c_since,
        @c_credit,
        @c_credit_lim,
        @c_discount,
        @c_balance,
        @screen_data
GO

CREATE PROCEDURE    tpcc_stocklevel
```

```
                        @w_id           int,
                        @d_id           tinyint,
                        @threshhold     smallint
AS
DECLARE @delaytime  varchar(30)

-------------------------------------------------
-- uniform random delay of 0 - 3.6 second; avg = 1.8
-------------------------------------------------
SELECT  @delaytime  = '00:00:0' +
CAST(CAST((RAND()*0.20) AS decimal(4,3)) AS char(5))

WAITFOR delay @delaytime

SELECT  49
GO

CREATE PROCEDURE tpcc_version

AS
DECLARE @version    char(8)

BEGIN
    SELECT  @version = '4.10.000'

    SELECT  @version AS 'Version'
END
GO

CREATE TABLE    order_line_null (
                                    [ol_i_id] [int]
NOT NULL ,
                                    [ol_supply_w_id]
[int] NOT NULL ,
                                    [ol_delivery_d]
[datetime] NOT NULL ,
                                    [ol_quantity]
[smallint] NOT NULL ,
                                    [ol_amount]
[numeric](6, 2) NOT NULL
) ON [PRIMARY]
GO

INSERT INTO order_line_null VALUES ( 101, 1,
GETDATE(), 1, 123.45 )
INSERT INTO order_line_null VALUES ( 102, 1,
GETDATE(), 2, 123.45 )
INSERT INTO order_line_null VALUES ( 103, 1,
GETDATE(), 3, 123.45 )
INSERT INTO order_line_null VALUES ( 104, 1,
GETDATE(), 4, 123.45 )
INSERT INTO order_line_null VALUES ( 105, 1,
GETDATE(), 5, 123.45 )
INSERT INTO order_line_null VALUES ( 106, 1,
GETDATE(), 1, 123.45 )
INSERT INTO order_line_null VALUES ( 107, 1,
GETDATE(), 2, 123.45 )
INSERT INTO order_line_null VALUES ( 108, 1,
GETDATE(), 3, 123.45 )
INSERT INTO order_line_null VALUES ( 109, 1,
GETDATE(), 4, 123.45 )
INSERT INTO order_line_null VALUES ( 110, 1,
GETDATE(), 5, 123.45 )
INSERT INTO order_line_null VALUES ( 111, 1,
GETDATE(), 1, 123.45 )
INSERT INTO order_line_null VALUES ( 112, 1,
GETDATE(), 2, 123.45 )
INSERT INTO order_line_null VALUES ( 113, 1,
GETDATE(), 3, 123.45 )
INSERT INTO order_line_null VALUES ( 114, 1,
GETDATE(), 4, 123.45 )
INSERT INTO order_line_null VALUES ( 115, 1,
GETDATE(), 5, 123.45 )
GO
```

ordstat.sql

```
------------------------------------------------------
-------------
--
--
--  File:   ORDSTAT.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates order status stored procedure
--
--
--
--          Interface Level:    4.20.000
--
--
--
------------------------------------------------------
-------------
SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

USE tpcc
GO

IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_orderstatus' )
    DROP PROCEDURE tpcc_orderstatus
GO

CREATE PROCEDURE    tpcc_orderstatus
                        @w_id       int,
                        @d_id       tinyint,


                        @c_id       int,
                        @c_last     char(16) = ''

AS
DECLARE @c_balance      money,
        @c_first        char(16),
        @c_middle       char(2),
        @o_id           int,
        @o_entry_d      datetime,
        @o_carrier_id   smallint,
        @cnt            smallint

BEGIN TRANSACTION o
    IF (@c_id = 0)
    BEGIN
        ------------------------------------------
        --  get customer id and info using last name
        ------------------------------------------
        SELECT  @cnt    = (count(*)+1)/2
        FROM    customer WITH (repeatableread)
        WHERE   c_last  = @c_last AND
                c_w_id  = @w_id AND
                c_d_id  = @d_id

        SET     rowcount @cnt

        SELECT  @c_id       = c_id,
                @c_balance  = c_balance,
                @c_first    = c_first,
                @c_last     = c_last,
                @c_middle   = c_middle
        FROM    customer WITH (repeatableread)
        WHERE   c_last      = @c_last AND
                c_w_id      = @w_id AND
                c_d_id      = @d_id
        ORDER   BY c_w_id, c_d_id, c_last, c_first

        SET rowcount 0
    END
    ELSE
    BEGIN
        ----------------------------
        --  get customer info if by id
        ----------------------------
        SELECT  @c_balance  = c_balance,
                @c_first    = c_first,
                @c_middle   = c_middle,
                @c_last     = c_last
        FROM    customer WITH (repeatableread)
        WHERE   c_id        = @c_id AND
                c_d_id      = @d_id AND
                c_w_id      = @w_id

        SELECT  @cnt          = @@rowcount
    END

    ----------------------
    --  if no such customer
    ----------------------
    IF (@cnt = 0)
    BEGIN
        RAISERROR('Customer not found',18,1)
        GOTO custnotfound
    END

    -----------------
    --  get order info
    -----------------
```

```
    SELECT  @o_id           = o_id,
            @o_entry_d      = o_entry_d,
            @o_carrier_id   = o_carrier_id

    FROM    orders WITH (serializable)
    WHERE   o_c_id          = @c_id AND
            o_d_id          = @d_id AND
            o_w_id          = @w_id
    ORDER   BY o_id ASC

    -------------------------------------------
    --  select order lines for the current order
    -------------------------------------------
    SELECT  ol_supply_w_id,
            ol_i_id,
            ol_quantity,
            ol_amount,
            ol_delivery_d
    FROM    order_line WITH (repeatableread)
    WHERE   ol_o_id = @o_id AND
            ol_d_id = @d_id AND
            ol_w_id = @w_id

    custnotfound:

    COMMIT TRANSACTION o

    -------------------------
    --  return data to client
    -------------------------
    SELECT  @c_id,
            @c_last,
            @c_first,
            @c_middle,

            @o_entry_d,
            @o_carrier_id,
            @c_balance,
            @o_id
GO
```

payment.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   PAYMENT.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates payment stored procedure
--
--
--
```

```
--          Interface Level:    4.20.000
--
--
--
-------------------------------------------------------
-------------
SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

USE tpcc
GO

IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_payment' )
    DROP PROCEDURE tpcc_payment
GO

CREATE PROCEDURE    tpcc_payment
                        @w_id       int,
                        @c_w_id     int,
                        @h_amount   smallmoney,
                        @d_id       tinyint,
                        @c_d_id     tinyint,
                        @c_id       int,
                        @c_last     char(16) = ""



AS
DECLARE @w_street_1     char(20),
        @w_street_2     char(20),
        @w_city         char(20),
        @w_state        char(2),
        @w_zip          char(9),
        @w_name         char(10),
        @d_street_1     char(20),
        @d_street_2     char(20),
        @d_city         char(20),
        @d_state        char(2),
        @d_zip          char(9),
        @d_name         char(10),
        @c_first        char(16),
        @c_middle       char(2),
        @c_street_1     char(20),
        @c_street_2     char(20),
        @c_city         char(20),
        @c_state        char(2),
        @c_zip          char(9),
        @c_phone        char(16),
        @c_since        datetime,
        @c_credit       char(2),
        @c_credit_lim   money,
        @c_balance      money,
        @c_discount     smallmoney,
        @c_data         char(42),
        @datetime       datetime,
        @w_ytd          money,
        @d_ytd          money,
        @cnt            smallint,
        @val            smallint,
        @screen_data    char(200),
        @d_id_local     tinyint,
        @w_id_local     int,
        @c_id_local     int

    SELECT  @screen_data    = ""

    BEGIN TRANSACTION p
        --  get payment date
        SELECT  @datetime = GETDATE()

        IF (@c_id = 0)
        BEGIN
            --  get customer id and info using last name
            SELECT  @cnt    = COUNT(*)
            FROM    customer WITH (repeatableread)
            WHERE   c_last  = @c_last AND
                                c_w_id  = @c_w_id AND
                                c_d_id  = @c_d_id



            SELECT  @val    = (@cnt + 1) / 2

            SET     rowcount @val

            SELECT  @c_id   = c_id
            FROM    customer WITH (repeatableread)
            WHERE   c_last  = @c_last AND
                                c_w_id  = @c_w_id AND
                                c_d_id  = @c_d_id
            ORDER   BY c_last, c_first

            SET     rowcount 0
        END

        --  get customer info and update balances

        UPDATE  customer
        SET     @c_balance      = c_balance = c_balance -
    @h_amount,
                c_payment_cnt   = c_payment_cnt + 1,
                c_ytd_payment   = c_ytd_payment +
    @h_amount,
                @c_first        = c_first,
                @c_middle       = c_middle,
                @c_last         = c_last,
                @c_street_1     = c_street_1,
                @c_street_2     = c_street_2,
                @c_city         = c_city,
                @c_state        = c_state,
                @c_zip          = c_zip,
                @c_phone        = c_phone,
                @c_credit       = c_credit,
                @c_credit_lim   = c_credit_lim,
                @c_discount     = c_discount,
                @c_since        = c_since,
                @c_id_local     = c_id
        WHERE   c_id            = @c_id AND
                c_w_id          = @c_w_id AND
                c_d_id          = @c_d_id

        --  if customer has bad credit get some more info
        IF (@c_credit = "BC")
```

```
    BEGIN
        --          compute new info
        SELECT  @c_data =   convert(char(5),@c_id) +
                            convert(char(4),@c_d_id)
+
                            convert(char(5),@c_w_id)
+
                            convert(char(4),@d_id) +
                            convert(char(5),@w_id) +

convert(char(19),@h_amount)

        --  update customer info
        UPDATE  customer
        SET     c_data          = @c_data +
substring(c_data, 1, 458),
                @screen_data    = @c_data +
substring(c_data, 1, 158)
        WHERE   c_id            = @c_id AND
                c_w_id          = @c_w_id AND
                c_d_id          = @c_d_id
    END

    --  get district data and update year-to-date
    UPDATE  district
    SET     d_ytd       = d_ytd + @h_amount,
            @d_street_1 = d_street_1,
            @d_street_2 = d_street_2,
            @d_city     = d_city,
            @d_state    = d_state,
            @d_zip      = d_zip,
            @d_name     = d_name,
            @d_id_local = d_id
    WHERE   d_w_id      = @w_id AND
            d_id        = @d_id

    --  get warehouse data and update year-to-date
    UPDATE  warehouse
    SET     w_ytd       = w_ytd + @h_amount,
            @w_street_1 = w_street_1,
            @w_street_2 = w_street_2,
            @w_city     = w_city,
            @w_state     = w_state,
            @w_zip      = w_zip,
            @w_name     = w_name,
            @w_id_local = w_id
    WHERE   w_id        = @w_id

    --  create history record
          INSERT INTO         history VALUES
(@c_id_local,
                              @c_d_id,
                              @c_w_id,
                              @d_id_local,
                              @w_id_local,
                              @datetime,
                              @h_amount,
                              @w_name + '    ' +
@d_name)

    COMMIT TRANSACTION p

    --  return data to client
    SELECT  @c_id,
            @c_last,
            @datetime,
            @w_street_1,
            @w_street_2,
            @w_city,
            @w_state,
            @w_zip,
            @d_street_1,
            @d_street_2,
            @d_city,
            @d_state,
            @d_zip,
            @c_first,
            @c_middle,
            @c_street_1,
            @c_street_2,
            @c_city,
            @c_state,
            @c_zip,
            @c_phone,
            @c_since,
            @c_credit,
            @c_credit_lim,
            @c_discount,
            @c_balance,
            @screen_data
GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO
```

random.c

```
//        File:               RANDOM.C
//                                        Microsoft
TPC-C Kit Ver. 4.62
//                                        Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000, 2001, 2002,
2005
//        Purpose:  Random number generation routines
for database loader


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

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

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

/*************************************************
************************
*
*
* random -
*
*       Implements a GOOD pseudo random number
generator.  This generator     *
*       will/should? run the complete period before
repeating.                *
*
*
* Copied from:
*
*       Random Numbers Generators: Good Ones Are Hard
to Find.                  *
*       Communications of the ACM - October 1988
Volume 31 Number 10          *
*
*
* Machine Dependencies:
*
*       long must be 2 ^ 31 - 1 or greater.
*
*
*
*************************************************
************************/

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

void seed(long val)
{

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

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

    Seed = val;

}


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

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

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

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

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

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

    return( Seed );
}


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

double drand()
{

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

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


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

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

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


          upper++;

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


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


    return rand_num;
}


#if 0

//Orginal code pgd 08/13/96

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

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

          upper++;

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


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


    return rand_num;
}
#endif

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

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

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

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

    return rand_num;
}
```

removedb.sql

```
USE master
```

```
GO
EXEC sp_dropdevice 'tpccback1'
GO
EXEC sp_dropdevice 'tpccback3'
GO
EXEC sp_dropdevice 'tpccback5'
GO
EXEC sp_dropdevice 'tpccback7'
GO
EXEC sp_dropdevice 'tpccback9'
GO
EXEC sp_dropdevice 'tpccback11'
GO
EXEC sp_dropdevice 'tpccback13'
GO
EXEC sp_dropdevice 'tpccback15'
GO
EXEC sp_dropdevice 'tpccback17'
GO
EXEC sp_dropdevice 'tpccback19'
GO
EXEC sp_dropdevice 'tpccback21'
GO
EXEC sp_dropdevice 'tpccback23'
GO
EXEC sp_dropdevice 'tpccback25'
GO
EXEC sp_dropdevice 'tpccback27'
GO
EXEC sp_dropdevice 'tpccback29'
GO
EXEC sp_dropdevice 'tpccback31'
GO
EXEC sp_dropdevice 'tpccback33'
GO
EXEC sp_dropdevice 'tpccback35'
GO
EXEC sp_dropdevice 'tpccback37'
GO
EXEC sp_dropdevice 'tpccback39'
GO
EXEC sp_dropdevice 'tpccback41'
GO
EXEC sp_dropdevice 'tpccback43'
GO
EXEC sp_dropdevice 'tpccback45'
GO
EXEC sp_dropdevice 'tpccback47'
GO
EXEC sp_dropdevice 'tpccback49'
GO
EXEC sp_dropdevice 'tpccback51'
GO
EXEC sp_dropdevice 'tpccback53'
GO
```

restore.cmd

```
osql -E -i restore.sql
```

restore.sql

```
-----------------------------------------------------
-------------
--
--
--  File:   RESTORE.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.61
--
--          Copyright Microsoft, 2005
--
--
--
-----------------------------------------------------
-------------

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
                    CONVERT(VARCHAR(30),@startdate,
21)

LOAD DATABASE tpcc FROM tpccback1, tpccback2,
tpccback3, tpccback4, tpccback5, tpccback6,
tpccback7, tpccback8, tpccback9, tpccback10,
tpccback11, tpccback12, tpccback13, tpccback14,
tpccback15, tpccback16 WITH stats = 1, replace

SELECT  @enddate = GETDATE()
SELECT  'End date: ',
                    CONVERT(VARCHAR(30),@enddate, 21)
SELECT  'Elapsed time (in seconds): ',
                    DATEDIFF(second, @startdate,
@enddate)
GO
```

restore_datalog.sql

```
-----------------------------------------------------
-------------
--
--
--  File:   RESTORE.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.61
--
--          Copyright Microsoft, 2005
--
--
--
-----------------------------------------------------
-------------

DECLARE @startdate  DATETIME,
        @enddate    DATETIME

SELECT  @startdate = GETDATE()
SELECT  'Start date:',
                    CONVERT(VARCHAR(30),@startdate,
21)

LOAD DATABASE tpcc FROM tpccback1, tpccback2,
tpccback3, tpccback4, tpccback5, tpccback6,
tpccback7, tpccback8, tpccback9, tpccback10,
tpccback11, tpccback12, tpccback13, tpccback14,
tpccback15, tpccback16 WITH stats = 1, replace,
norecovery

SELECT  @enddate = GETDATE()
SELECT  'End date: ',
        CONVERT(VARCHAR(30),@enddate, 21)
SELECT  'Elapsed time (in seconds): ',
        DATEDIFF(second, @startdate, @enddate)
GO
```

start-sql.cmd

```
cd C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\Binn
sqlservr.exe -c -x -T3502 -T8011 -T8012 -T8018 -T8019
-T8710 -T661 -T834
```

StockLev.sql

```
-----------------------------------------------------
-------------
--
--
--  File:   STOCKLEV.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates stock level stored procedure
--
--
--
--          Interface Level:    4.20.000
--
--
--
-----------------------------------------------------
-------------
SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO

USE tpcc
GO
```

```
IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_stocklevel' )
    DROP PROCEDURE tpcc_stocklevel
GO

CREATE PROCEDURE    tpcc_stocklevel
                        @w_id           int,
                        @d_id           tinyint,
                        @threshhold     smallint

AS
DECLARE @o_id_low   int,
        @o_id_high  int

SELECT  @o_id_low   = (d_next_o_id - 20),
        @o_id_high  = (d_next_o_id - 1)
FROM    district
WHERE   d_w_id      = @w_id AND
        d_id        = @d_id

SELECT  COUNT(DISTINCT(s_i_id))
FROM    stock,
        order_line
WHERE   ol_w_id     = @w_id AND
        ol_d_id     = @d_id and
        ol_o_id     BETWEEN @o_id_low AND
                            @o_id_high AND
        s_w_id      = ol_w_id AND
        s_i_id      = ol_i_id AND
        s_quantity  < @threshhold
OPTION(ORDER GROUP)
GO

SET QUOTED_IDENTIFIER OFF
GO

SET ANSI_NULLS ON
GO
```

stop-sql.cmd

```
sqlcmd -A -E -Q "shutdown"
```

strings.c

```
//        File:               STRINGS.C
//                                    Microsoft
TPC-C Kit Ver. 4.51
//                                    Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000, 2001, 2002,
2003
//        Purpose:  Source file for database loader
string functions


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

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

void MakeAddress(char *street_1,
                                            char
*street_2,
                                            char *city,
                                            char *state,
                                            char *zip)
{

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

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

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

    return;

}


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

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

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

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

                if (strlen(name) < LAST_NAME_LEN)
                {

          PaddString(LAST_NAME_LEN, name);
                }

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


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

    return;

}


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

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

```
//It is completely unreasonable to stuff non-printing
chars into the text fields.
//-CLevine 08/13/96

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

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

          len= RandomNumber(x, y);

          for (i=0; i<len; i++)
                    str[i] =
chArray[RandomNumber(0,chArrayMax)];
          str[len] = 0;

    return len;
}

int MakeAlphaStringPadded( int  minLen, int  maxLen,
int  padLen, char *str)
{
          int                 len;
          int                 i;
    char    cc = 'a';
          static    char chArray[] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnop
qrstuvwxyz";
          static    int       chArrayMax = 61;

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

          len= RandomNumber(minLen, maxLen);

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



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

int MakeOriginalAlphaString(int  x,

              int  y,

                    int  z,

              char *str,

            int  percent)
{
    int             len;
    int             val;
    int             start;

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

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

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

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

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


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

    return len;
}



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

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

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

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

          str[16] = 0;

    return 16;
}


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

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

          strcpy(str, "000011111");

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

    return 9;
}


//=================================================
==================
//
// Function name: InitString
//
//=================================================
==================
void InitString(char *str, int len)
{
```

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

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


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

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

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

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

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


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

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

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

          return;
}
```

tables.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   TABLES.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Creates TPC-C tables
--
-------------------------------------------------------
-------------

SET ANSI_NULL_DFLT_OFF ON
GO

USE tpcc
GO

-----------------------------------
-- Remove all existing TPC-C tables
-----------------------------------
if exists ( select name from sysobjects where name =
'warehouse' )
          drop table warehouse
go
if exists ( select name from sysobjects where name =
'district' )
          drop table district
go
if exists ( select name from sysobjects where name =
'customer' )
          drop table customer
go
if exists ( select name from sysobjects where name =
'history' )
          drop table history
go
if exists ( select name from sysobjects where name =
'new_order' )
          drop table new_order
go
if exists ( select name from sysobjects where name =
'orders' )
          drop table orders
go
if exists ( select name from sysobjects where name =
'order_line' )
          drop table order_line
go
if exists ( select name from sysobjects where name =
'item' )
          drop table item
go
if exists ( select name from sysobjects where name =
'stock' )
          drop table stock
go

--------------------
-- Create new tables
--------------------
create table warehouse
(
          w_id                int,
          w_ytd               money,
          w_tax               smallmoney,
          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)
) on MSSQL_fg
go

create table district
(
          d_id                tinyint,
          d_w_id              int,
          d_ytd               money,
          d_next_o_id                 int,

          d_tax               smallmoney,
          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)
) on MSSQL_fg
go

create table customer
(
          c_id                int,
          c_d_id              tinyint,
          c_w_id              int,
          c_discount          smallmoney,
          c_credit_lim        money,
          c_last              char(16),
          c_first             char(16),
          c_credit            char(2),
          c_balance           money,
          c_ytd_payment       money,
          c_payment_cnt       smallint,
          c_delivery_cnt      smallint,
          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_middle            char(2),
          c_data              char(500)
) on MSSQL_fg
```

```
go

-- Use the following table option if using c_data
varchar(max)
-- sp_tableoption 'customer','large value types out
of row','1'
-- go

create table history
(
          h_c_id              int,
          h_c_d_id            tinyint,
          h_c_w_id            int,
          h_d_id              tinyint,
          h_w_id              int,
          h_date              datetime,
          h_amount            smallmoney,
          h_data              char(24)
) on MSSQL_fg
go

create table new_order
(
          no_o_id             int,
          no_d_id             tinyint,
          no_w_id             int
) on MSSQL_fg
go

create table orders
(
          o_id                int,
          o_d_id              tinyint,
          o_w_id              int,
          o_c_id              int,
          o_carrier_id        tinyint,
          o_ol_cnt            tinyint,
          o_all_local         tinyint,
          o_entry_d           datetime
) on MSSQL_fg
go

create table order_line
(
          ol_o_id             int,
          ol_d_id             tinyint,
          ol_w_id             int,
          ol_number           tinyint,
          ol_i_id             int,
          ol_delivery_d       datetime,
          ol_amount           smallmoney,
          ol_supply_w_id      int,
          ol_quantity         smallint,
          ol_dist_info        char(24)
) on MSSQL_fg
go

create table item
(
          i_id                int,
          i_name              char(24),
          i_price             smallmoney,
          i_data              char(50),
          i_im_id             int
) on MSSQL_fg
go

create table stock
(
          s_i_id              int,
          s_w_id              int,
          s_quantity          smallint,
          s_ytd               int,
          s_order_cnt         smallint,
          s_remote_cnt        smallint,
          s_data              char(50),
          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)
) on MSSQL_fg
go
```

time.c

```
//        File:               TIME.C
//                                          Microsoft
TPC-C Kit Ver. 4.62
//                                          Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000, 2001, 2002,
2005
//        Purpose:  Source file for time functions


// Includes
#include "tpcc.h"

// Globals
static long start_sec;


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

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

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

   _ftime(&el_time);

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

   return time_now;
}
```

tpcc.h

```
//        File:               TPCC.H
//                                          Microsoft
TPC-C Kit Ver. 4.51
//                                          Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000, 2001, 2002,
2003, 2005
//        Purpose:  Header file for TPC-C database
loader


// Build number of TPC Benchmark Kit
#define TPCKIT_VER  "4.51"

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

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

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

// Default environment constants
#define SERVER                                       ""
```

```
#define DATABASE
        "tpcc"
#define USER
        "sa"
#define PASSWORD                            ""

// Default loader arguments
#define BATCH
                10000
#define DEFLDPACKSIZE
        32768
#define LOADER_RES_FILE
        "C:\\MSTPCC.450\\SETUP\\LOGS\\load.out"
#define LOADER_LOG_PATH
        "C:\\MSTPCC.450\\SETUP\\LOGS\\"
#define LOADER_NURAND_C
        123
#define DEF_STARTING_WAREHOUSE                  1
#define BUILD_INDEX
        1       // build both data and indexes
#define INDEX_ORDER
        1       // build indexes before load
#define SCALE_DOWN
        0   // build a normal scale database
#define INDEX_SCRIPT_PATH
        "scripts"

typedef struct
{
    char
        *server;
    char
        *database;
    char
        *user;
    char
        *password;
        BOOL
        tables_all;                             //
set if loading all tables
        BOOL
        table_item;                             //
set if loading ITEM table specifically
        BOOL
        table_warehouse;    // set if loading
WAREHOUSE, DISTRICT, and STOCK
        BOOL
        table_customer;             // set if
loading CUSTOMER and HISTORY
        BOOL
        table_orders;               // set if
loading NEW-ORDER, ORDERS, ORDER-LINE
    long
        num_warehouses;
    long
        batch;
    long
        verbose;
        long
        pack_size;
        char
        *loader_res_file;
        char
        *log_path;
        char
        *synch_servername;
        long
        case_sensitivity;
        long
        starting_warehouse;
        long
        build_index;
        long
        index_order;
        long
        scale_down;
        char
        *index_script_path;
} TPCCLDR_ARGS;

// String length constants
        #define SERVER_NAME_LEN          20
        #define DATABASE_NAME_LEN        20
        #define USER_NAME_LEN            20
        #define PASSWORD_LEN             20
        #define TABLE_NAME_LEN           20
        #define I_DATA_LEN               50
        #define I_NAME_LEN               24
        #define BRAND_LEN                 1
        #define LAST_NAME_LEN            16
        #define W_NAME_LEN               10
        #define ADDRESS_LEN              20
        #define STATE_LEN                 2
        #define ZIP_LEN                   9
        #define S_DIST_LEN               24
        #define S_DATA_LEN               50
        #define D_NAME_LEN               10
        #define FIRST_NAME_LEN           16
        #define MIDDLE_NAME_LEN           2
        #define PHONE_LEN                16
        #define CREDIT_LEN                2
        #define C_DATA_LEN              500
        #define H_DATA_LEN               24
        #define DIST_INFO_LEN            24
        #define MAX_OL_NEW_ORDER_ITEMS   15
        #define MAX_OL_ORDER_STATUS_ITEMS 15
        #define STATUS_LEN               25
        #define OL_DIST_INFO_LEN
24
        #define C_SINCE_LEN
           23
        #define H_DATE_LEN
           23
        #define OL_DELIVERY_D_LEN
23
        #define O_ENTRY_D_LEN            23


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

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

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

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

tpccldr.c

```
//=================================================
==================
//        File:               TPCCLDR.C
//                                        Microsoft
TPC-C Kit Ver. 4.51
//                                        Copyright
Microsoft, 1996, 1997, 1998, 1999,
//                                        2000, 2001,
2002, 2003
//        Purpose:  Source file for TPC-C database
loader
//=================================================
==================
// Includes
#include "tpcc.h"
#include "search.h"

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

// Functions declarations
void HandleErrorDBC (SQLHDBC  hdbc1);
long NURand();
void LoadItem();
void LoadWarehouse();
void Stock();
void District();
void LoadCustomer();
```

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

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

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

typedef struct
{
    long                                    c_id;
    short                                   c_d_id;
          long
          c_w_id;
          char
          c_first[FIRST_NAME_LEN+1];
    char
          c_middle[MIDDLE_NAME_LEN+1];
    char
          c_last[LAST_NAME_LEN+1];
    char
          c_street_1[ADDRESS_LEN+1];
    char
          c_street_2[ADDRESS_LEN+1];
    char
          c_city[ADDRESS_LEN+1];
    char
          c_state[STATE_LEN+1];
    char
          c_zip[ZIP_LEN+1];
    char
          c_phone[PHONE_LEN+1];
    char
          c_credit[CREDIT_LEN+1];
    double
          c_credit_lim;
    double
          c_discount;
          char
          c_balance[6];
    double
          c_ytd_payment;
          short
          c_payment_cnt;
          short
          c_delivery_cnt;
          char
          c_data[C_DATA_LEN+1];
          double
          h_amount;
    char
          h_data[H_DATA_LEN+1];
} CUSTOMER_STRUCT;

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

typedef struct
{
    long                time_start;
} LOADER_TIME_STRUCT;


// Global variables
char      szLastError[300];

HENV      henv;

HDBC      v_hdbc;
                    // for SQL Server version
verification
HDBC      i_hdbc1;
          // for ITEM table
HDBC      w_hdbc1;
          // for WAREHOUSE, DISTRICT, STOCK
HDBC      c_hdbc1;
          // for CUSTOMER
HDBC      c_hdbc2;
          // for HISTORY
HDBC      o_hdbc1;
          // for ORDERS
HDBC      o_hdbc2;
          // for NEW-ORDER

HDBC      o_hdbc3;
          // for ORDER-LINE

HSTMT     v_hstmt;
          // for SQL Server version verification
HSTMT     i_hstmt1;
HSTMT     w_hstmt1;
HSTMT     c_hstmt1, c_hstmt2;
HSTMT     o_hstmt1, o_hstmt2, o_hstmt3;

int                 total_db_errors;

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

TPCCLDR_ARGS    *aptr, args;

//================================================
==================
//
// Function name: main
//
//================================================
==================
int main(int  argc, char **argv)
{

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

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


printf("\n*****************************************
********");
          printf("\n*
*");
```

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

    // process command line arguments
    aptr = &args;
    GetArgsLoader(argc, argv, aptr);

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

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

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

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

        // open connections to SQL Server
        OpenConnections();

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

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

    // start loading data
    sprintf(buffer,"TPC-C load started for %ld
warehouses.\n",aptr->num_warehouses);
        if      (aptr->scale_down == 1)
        {
                sprintf(buffer,"SCALED DOWN
DATABASE.\n");
        }

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

        main_time_start = (TimeNow() / MILLI);

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


                hThread[0] = CreateThread(NULL,

                                0,


(LPTHREAD_START_ROUTINE) LoadItem,

                                NULL,

                                0,

                                &dwThreadID[0]);

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

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

                hThread[1] = CreateThread(NULL,

                                0,


(LPTHREAD_START_ROUTINE) LoadWarehouse,

                                NULL,

                                0,

                                &dwThreadID[1]);

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

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

                hThread[2] = CreateThread(NULL,

                                0,


(LPTHREAD_START_ROUTINE) LoadCustomer,

                                NULL,

                                0,

                                &dwThreadID[2]);

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

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

                hThread[3] = CreateThread(NULL,

                                0,


(LPTHREAD_START_ROUTINE) LoadOrders,

                                NULL,

                                0,

                                &dwThreadID[3]);

                if (hThread[3] == NULL)
                {
```

```
                        printf("Error, failed
in creating creating main thread = 3.\n");
                        exit(-1);
                }
        }

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

        CloseHandle(hThread[i]);
                        hThread[i] = NULL;
                }
        }

        main_time_end = (TimeNow() / MILLI);

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

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

        fclose(fLoader);

        SQLFreeEnv(henv);

    exit(0);

        return 0;
}


//=================================================
==================
//
// Function name: LoadItem
//
//=================================================
==================
void LoadItem()
{
    int                 i;
        long            i_id;
        long            i_im_id;
    char            i_name[I_NAME_LEN+1];
    double              i_price;
    char            i_data[I_DATA_LEN+1];
        char            name[20];
        long            time_start;
        RETCODE         rc;
        DBINT           rcint;
        char            bcphint[128];
        char            err_log_path[256];

        // Seed with unique number
        seed(11);

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

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

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

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

        strcpy(err_log_path,aptr->log_path);
        strcat(err_log_path,"item.err");
        rc = bcp_init(i_hdbc1, name, NULL,
err_log_path , DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

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

        HandleErrorDBC(i_hdbc1);
        }

        i = 0;
        rc = bcp_bind(i_hdbc1, (BYTE *) &i_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);
        rc = bcp_bind(i_hdbc1, (BYTE *) i_name, 0,
I_NAME_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);
        rc = bcp_bind(i_hdbc1, (BYTE *) &i_price,
0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);
        rc = bcp_bind(i_hdbc1, (BYTE *) i_data, 0,
SQL_VARLEN_DATA, "", 1, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);
        rc = bcp_bind(i_hdbc1, (BYTE *) &i_im_id,
0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(i_hdbc1);

        time_start = (TimeNow() / MILLI);

        item_rows_loaded = 0;

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

                MakeAlphaStringPadded(14, 24,
I_NAME_LEN, i_name);

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

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

                rc = bcp_sendrow(i_hdbc1);

                if (rc != SUCCEED)

        HandleErrorDBC(i_hdbc1);

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

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


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


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

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

}

//=================================================
==========================
//
// Function   : LoadWarehouse
//
// Loads WAREHOUSE table and loads Stock and District
as Warehouses are created
//
//=================================================
==========================
void LoadWarehouse()
{
        int                 i;
        long        w_id;
    char    w_name[W_NAME_LEN+1];
    char    w_street_1[ADDRESS_LEN+1];
    char    w_street_2[ADDRESS_LEN+1];
    char    w_city[ADDRESS_LEN+1];
    char    w_state[STATE_LEN+1];
    char    w_zip[ZIP_LEN+1];
    double          w_tax;
    double          w_ytd;
        char        name[20];
```

```
        long      time_start;
        RETCODE   rc;
        DBINT     rcint;
        char      bcphint[128];
        char      err_log_path[256];

        // Seed with unique number
        seed(2);

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

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

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

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

        strcpy(err_log_path,aptr->log_path);
        strcat(err_log_path,"whouse.err");
        rc = bcp_init(w_hdbc1, name, NULL,
err_log_path, DB_IN);

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

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

        HandleErrorDBC(w_hdbc1);
        }

        i = 0;
        rc = bcp_bind(w_hdbc1, (BYTE *) &w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) &w_ytd, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) &w_tax, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) w_name, 0,
W_NAME_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) w_street_1,
0, ADDRESS_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) w_street_2,
0, ADDRESS_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) w_city, 0,
ADDRESS_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) w_state, 0,
STATE_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) w_zip, 0,
ZIP_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        time_start = (TimeNow() / MILLI);

        warehouse_rows_loaded = 0;

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

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

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

                w_ytd = 300000.00;

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

        HandleErrorDBC(w_hdbc1);

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

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


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

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

        stock_rows_loaded = 0;
        district_rows_loaded = 0;

        District();
        Stock();
}

//================================================
==================
//
// Function   : District
//
//================================================
==================
void District()
{
    int             i;
          short     d_id;
          long      d_w_id;
    char  d_name[D_NAME_LEN+1];
    char  d_street_1[ADDRESS_LEN+1];
    char  d_street_2[ADDRESS_LEN+1];
    char  d_city[ADDRESS_LEN+1];
    char  d_state[STATE_LEN+1];
    char  d_zip[ZIP_LEN+1];
    double          d_tax;
    double          d_ytd;
          char      name[20];
    long  d_next_o_id;
          long      time_start;
          long      w_id;
          RETCODE   rc;
          DBINT     rcint;
          char      bcphint[128];
          char      err_log_path[256];

          // Seed with unique number
          seed(4);

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

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

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

          strcpy(err_log_path,aptr->log_path);
          strcat(err_log_path,"district.err");
          rc = bcp_init(w_hdbc1, name, NULL,
err_log_path, DB_IN);
          if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

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

```
                if (rc != SUCCEED)

        HandleErrorDBC(w_hdbc1);
        }

        i = 0;
        rc = bcp_bind(w_hdbc1, (BYTE *) &d_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) &d_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) &d_ytd, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *)
&d_next_o_id, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT4,
++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) &d_tax, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) d_name, 0,
D_NAME_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) d_street_1,
0, ADDRESS_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) d_street_2,
0, ADDRESS_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) d_city, 0,
ADDRESS_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) d_state, 0,
STATE_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) d_zip, 0,
ZIP_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        d_ytd  = 30000.0;

        d_next_o_id = orders_per_district+1;

        time_start = (TimeNow() / MILLI);

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

                for (d_id = 1; d_id <=
DISTRICT_PER_WAREHOUSE; d_id++)
                {

        MakeAlphaStringPadded(6,10,D_NAME_LEN,
d_name);

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

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

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

        HandleErrorDBC(w_hdbc1);

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

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


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

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

    return;
}

//=================================================
====================
//
// Function   : Stock
//
//=================================================
====================
void Stock()
{
        int             i;
        long    s_i_id;
        long    s_w_id;
    short s_quantity;
    char  s_dist_01[S_DIST_LEN+1];
    char  s_dist_02[S_DIST_LEN+1];
    char  s_dist_03[S_DIST_LEN+1];
    char  s_dist_04[S_DIST_LEN+1];
    char  s_dist_05[S_DIST_LEN+1];
    char  s_dist_06[S_DIST_LEN+1];
    char  s_dist_07[S_DIST_LEN+1];
    char  s_dist_08[S_DIST_LEN+1];
    char  s_dist_09[S_DIST_LEN+1];
    char  s_dist_10[S_DIST_LEN+1];
    long  s_ytd;
    short s_order_cnt;
    short s_remote_cnt;
    char  s_data[S_DATA_LEN+1];
    short len;
        char    name[20];
        long    time_start;
        RETCODE rc;
        DBINT   rcint;
        char    bcphint[128];
        char    err_log_path[256];

        // Seed with unique number
        seed(3);

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

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

        strcpy(err_log_path,aptr->log_path);
        strcat(err_log_path,"stock.err");
        rc = bcp_init(w_hdbc1, name, NULL,
err_log_path, DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

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

        HandleErrorDBC(w_hdbc1);
        }

        i = 0;
        rc = bcp_bind(w_hdbc1, (BYTE *) &s_i_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) &s_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *)
&s_quantity, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
        rc = bcp_bind(w_hdbc1, (BYTE *) &s_ytd, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);
```

```
          rc = bcp_bind(w_hdbc1, (BYTE *)
&s_order_cnt, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *)
&s_remote_cnt, 0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_data, 0,
SQL_VARLEN_DATA, "", 1, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_01,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_02,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_03,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_04,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_05,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_06,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_07,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_08,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_09,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);
          rc = bcp_bind(w_hdbc1, (BYTE *) s_dist_10,
0, S_DIST_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(w_hdbc1);

          s_ytd = s_order_cnt = s_remote_cnt = 0;

          time_start = (TimeNow() / MILLI);

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

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

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

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

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

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

          HandleErrorDBC(w_hdbc1);

                              stock_rows_loaded++;

          CheckForCommit_Big(w_hdbc1, w_hstmt1,
stock_rows_loaded, "stock", &time_start);
                    }
          }

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


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

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

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

          return;
}

//=================================================
==================
//
// Function   : LoadCustomer
//
//=================================================
==================
void LoadCustomer()
{
          LOADER_TIME_STRUCT
customer_time_start;
          LOADER_TIME_STRUCT      history_time_start;
          long
          w_id;
    short                   d_id;
          DWORD
          dwThreadID[MAX_CUSTOMER_THREADS];
          HANDLE
          hThread[MAX_CUSTOMER_THREADS];
          char                            name[20];
          RETCODE
          rc;
          DBINT
          rcint;
          char
          bcphint[128];
          char
          cmd[256];
          int
                    num_procs;
          char
          err_log_path_cust[256];
          char
          err_log_path_hist[256];

          // Seed with unique number
          seed(5);

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

          // if build index before load...
          if ((aptr->build_index == 1) && (aptr-
>index_order == 1))
          {
                    BuildIndex("idxcuscl");
                     // check the number of
processors on this system
                    // if 8 or more processors, then
build index on History.
                     // if less than 8 processors, do
not build the index
                    num_procs = atoi(getenv(
"NUMBER_OF_PROCESSORS" ));
                    if ( num_procs >= 8 )
                              BuildIndex("idxhiscl");
          }

          // Initialize bulk copy
```

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

          strcpy(err_log_path_cust,aptr->log_path);
          strcat(err_log_path_cust,"customer.err");
          rc = bcp_init(c_hdbc1, name, NULL,
err_log_path_cust, DB_IN);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

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

          HandleErrorDBC(c_hdbc1);
          }

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

          rc = bcp_init(c_hdbc2, name, NULL,
"logs\\history.err", DB_IN);
          strcpy(err_log_path_hist,aptr->log_path);
          strcat(err_log_path_hist,"history.err");
          rc = bcp_init(c_hdbc2, name, NULL,
err_log_path_hist, DB_IN);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);

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

          customer_rows_loaded    = 0;
          history_rows_loaded     = 0;

          CustomerBufInit();

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

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

                              CustomerBufLoad(d_id,
w_id);

                              // Start parallel
loading threads here...

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

                              hThread[0] =
CreateThread(NULL,

                                        0,


(LPTHREAD_START_ROUTINE) LoadCustomerTable,


&customer_time_start,

                                        0,


&dwThreadID[0]);

                              if (hThread[0] == NULL)
                              {

                    printf("Error, failed in creating creating
thread = 0.\n");
                                        exit(-1);
                              }

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

                              hThread[1] =
CreateThread(NULL,

                                        0,


(LPTHREAD_START_ROUTINE) LoadHistoryTable,


&history_time_start,

                                        0,


&dwThreadID[1]);

                              if (hThread[1] == NULL)
                              {

                    printf("Error, failed in creating creating
thread = 1.\n");
                                        exit(-1);
                              }

                              WaitForSingleObject(
hThread[0], INFINITE );

                              WaitForSingleObject(
hThread[1], INFINITE );

                              if
(CloseHandle(hThread[0]) == FALSE)
                              {

                    printf("Error, failed in closing customer
thread handle with errno: %d\n", GetLastError());
                              }

                              if
(CloseHandle(hThread[1]) == FALSE)
                              {

                    printf("Error, failed in closing history
thread handle with errno: %d\n", GetLastError());
                              }
                    }
          }

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


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


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

          // if build index after load...
          if ((aptr->build_index == 1) && (aptr-
>index_order == 0))
          {
                    BuildIndex("idxcuscl");
                    // check the number of processors
on this system
                    // if 8 or more processors, then
build index on History.
                    // if less than 8 processors, do
not build the index
                    num_procs = atoi(getenv(
"NUMBER_OF_PROCESSORS" ));
                    if (num_procs >= 8)
                              BuildIndex("idxhiscl");
          }

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

          // Output the NURAND used for the loader
into C_FIRST for C_ID = 1,
          // C_W_ID = 1, and C_D_ID = 1
          sprintf(cmd, "osql -S%s -U%s -P%s -d%s -e -
Q\"update customer set c_first = 'C_LOAD = %d' where
c_id = 1 and c_w_id = 1 and c_d_id = 1\" >
%snurand_load.log",
```

```
                                        aptr->server,
                                        aptr->user,
                                        aptr-
>password,
                                        aptr-
>database,

          LOADER_NURAND_C,
                                        aptr-
>log_path);

          system(cmd);

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

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

    return;
}

//================================================
==================
//
// Function   : CustomerBufInit
//
//================================================
==================
void CustomerBufInit()
{
          long     i;

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


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

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

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

          strcpy(customer_buf[i].c_street_1,"");

          strcpy(customer_buf[i].c_street_2,"");

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

          strcpy(customer_buf[i].c_state,"");
                    strcpy(customer_buf[i].c_zip,"");

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

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

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


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

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


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


                    customer_buf[i].h_amount = 0;


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

//================================================
==================
//
// Function   : CustomerBufLoad
//
// Fills shared buffer for HISTORY and CUSTOMER
//================================================
==================
void CustomerBufLoad(int d_id, long w_id)
{
          long                                    i;
    CUSTOMER_SORT_STRUCT
c[CUSTOMERS_PER_DISTRICT];

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

          LastName(NURand(255,0,999,LOADER_NURAND_C),
c[i].c_last);


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

                    c[i].c_id = i+1;
          }

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

          for (i=0;i<customers_per_district;i++)
          {
                    customer_buf[i].c_d_id = d_id;
                    customer_buf[i].c_w_id = w_id;
                    customer_buf[i].h_amount = 10.0;
                    customer_buf[i].c_ytd_payment  =
10.0;
                    customer_buf[i].c_payment_cnt  =
1;
                    customer_buf[i].c_delivery_cnt =
0;
                    customer_buf[i].c_id = c[i].c_id;
                    strcpy(customer_buf[i].c_first,
c[i].c_first);
                    strcpy(customer_buf[i].c_last,
c[i].c_last);
                    customer_buf[i].c_middle[0] =
'O';
                    customer_buf[i].c_middle[1] =
'E';

          MakeAddress(customer_buf[i].c_street_1,
customer_buf[i].c_street_2,
customer_buf[i].c_city,
customer_buf[i].c_state,
customer_buf[i].c_zip);
                    MakeNumberString(16, 16,
PHONE_LEN, customer_buf[i].c_phone);

                    if (RandomNumber(1L, 100L) > 10)

          customer_buf[i].c_credit[0] = 'G';
                    else

          customer_buf[i].c_credit[0] = 'B';
                    customer_buf[i].c_credit[1] =
'C';
                    customer_buf[i].c_credit_lim =
50000.0;
                    customer_buf[i].c_discount =
((float) RandomNumber(0L, 5000L)) / 10000.0;


          strcpy(customer_buf[i].c_balance,"-10.0");
                    MakeAlphaStringPadded(300, 500,
C_DATA_LEN, customer_buf[i].c_data);

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

//================================================
==================
//
// Function   : LoadCustomerTable
//
//================================================
==================
void LoadCustomerTable(LOADER_TIME_STRUCT
*customer_time_start)
{
```

```
        long                i;
    long        c_id;
    short       c_d_id;
        long                c_w_id;
    char        c_first[FIRST_NAME_LEN+1];
    char        c_middle[MIDDLE_NAME_LEN+1];
    char        c_last[LAST_NAME_LEN+1];
    char        c_street_1[ADDRESS_LEN+1];
    char        c_street_2[ADDRESS_LEN+1];
    char        c_city[ADDRESS_LEN+1];
    char        c_state[STATE_LEN+1];
    char        c_zip[ZIP_LEN+1];
    char        c_phone[PHONE_LEN+1];
    char        c_credit[CREDIT_LEN+1];
    double      c_credit_lim;
    double      c_discount;
        char                c_balance[6];
    double      c_ytd_payment;
    short       c_payment_cnt;
    short       c_delivery_cnt;
    char        c_data[C_DATA_LEN+1];
        char                c_since[C_SINCE_LEN+1];

        RETCODE             rc;

    i = 0;
        rc = bcp_bind(c_hdbc1, (BYTE *) &c_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) &c_d_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
        rc = bcp_bind(c_hdbc1, (BYTE *) &c_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) &c_discount, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) &c_credit_lim, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_last, 0,
LAST_NAME_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
        rc = bcp_bind(c_hdbc1, (BYTE *) c_first, 0,
FIRST_NAME_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_credit, 0,
CREDIT_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_balance, 0, 5,
NULL, 0, SQLCHARACTER, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) &c_ytd_payment,
0, SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) &c_payment_cnt,
0, SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *)
&c_delivery_cnt,0, SQL_VARLEN_DATA, NULL, 0, SQLINT2,
++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_street_1, 0,
ADDRESS_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_street_2, 0,
ADDRESS_LEN, NULL,0,0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_city, 0,
ADDRESS_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_state, 0,
STATE_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_zip, 0,
ZIP_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_phone, 0,
PHONE_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
        rc = bcp_bind(c_hdbc1, (BYTE *) &c_since,
0, C_SINCE_LEN, NULL, 0, SQLCHARACTER, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
        rc = bcp_bind(c_hdbc1, (BYTE *) c_middle,
0, MIDDLE_NAME_LEN,NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *) c_data, 0,
C_DATA_LEN, NULL, 0, 0, ++i);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc1);

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

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

                FormatDate(&c_since);


                c_credit_lim =
customer_buf[i].c_credit_lim;
                c_discount =
customer_buf[i].c_discount;
                strcpy(c_balance,
customer_buf[i].c_balance);
                c_ytd_payment =
customer_buf[i].c_ytd_payment;
                c_payment_cnt =
customer_buf[i].c_payment_cnt;
                c_delivery_cnt =
customer_buf[i].c_delivery_cnt;
                strcpy(c_data,
customer_buf[i].c_data);

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

        HandleErrorDBC(c_hdbc1);

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

//================================================
==================
//
// Function   : LoadHistoryTable
//
//================================================
==================
void LoadHistoryTable(LOADER_TIME_STRUCT
*history_time_start)
{
        long                i;
    long        c_id;
    short       c_d_id;
        long                c_w_id;
        double    h_amount;
    char        h_data[H_DATA_LEN+1];
```

```
          char                h_date[H_DATE_LEN+1];

          RETCODE             rc;

          i = 0;
    rc = bcp_bind(c_hdbc2, (BYTE *) &c_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);
    rc = bcp_bind(c_hdbc2, (BYTE *) &c_d_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);
          rc = bcp_bind(c_hdbc2, (BYTE *) &c_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);
    rc = bcp_bind(c_hdbc2, (BYTE *) &c_d_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);
          rc = bcp_bind(c_hdbc2, (BYTE *) &c_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);
          rc = bcp_bind(c_hdbc2, (BYTE *) &h_date, 0,
H_DATE_LEN, NULL, 0, SQLCHARACTER, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);
    rc = bcp_bind(c_hdbc2, (BYTE *) &h_amount, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);
    rc = bcp_bind(c_hdbc2, (BYTE *) h_data, 0,
H_DATA_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);

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


                    FormatDate(&h_date);


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


          HandleErrorDBC(c_hdbc2);


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

//=================================================
=========-==============
//
// Function   : LoadOrders
//
//=================================================
=========================
void LoadOrders()
{
          LOADER_TIME_STRUCT      orders_time_start;
          LOADER_TIME_STRUCT
new_order_time_start;
          LOADER_TIME_STRUCT
order_line_time_start;
          long
          w_id;
    short                     d_id;
          DWORD
          dwThreadID[MAX_ORDER_THREADS];
          HANDLE
          hThread[MAX_ORDER_THREADS];
          char                          name[20];
          RETCODE
          rc;
          char
          bcphint[128];
          char
          err_log_path_ord[256];
          char
          err_log_path_nord[256];
          char
          err_log_path_ordl[256];

          // seed with unique number
          seed(6);

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

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

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

          rc = bcp_init(o_hdbc1, name, NULL,
"logs\\orders.err", DB_IN);
          strcpy(err_log_path_ord,aptr->log_path);
          strcat(err_log_path_ord,"orders.err");
          rc = bcp_init(o_hdbc1, name, NULL,
err_log_path_ord, DB_IN);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

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

          HandleErrorDBC(o_hdbc1);
          }

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

          rc = bcp_init(o_hdbc2, name, NULL,
"logs\\neword.err", DB_IN);
          strcpy(err_log_path_nord,aptr->log_path);
          strcat(err_log_path_nord,"neword.err");
          rc = bcp_init(o_hdbc2, name, NULL,
err_log_path_nord, DB_IN);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);

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

          HandleErrorDBC(o_hdbc2);
          }

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

          rc = bcp_init(o_hdbc3, name, NULL,
"logs\\ordline.err", DB_IN);
          strcpy(err_log_path_ordl,aptr->log_path);
          strcat(err_log_path_ordl,"ordline.err");
          rc = bcp_init(o_hdbc3, name, NULL,
err_log_path_ordl, DB_IN);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

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

          HandleErrorDBC(o_hdbc3);
          }
```

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

        OrdersBufInit();

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

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

                        OrdersBufLoad(d_id,
w_id);

                        // start parallel
loading threads here...
                        // start Orders table
thread
                        printf("...Loading
Order Table for: d_id = %d, w_id = %d\n", d_id,
w_id);

                        hThread[0] =
CreateThread(NULL,

                                        0,


(LPTHREAD_START_ROUTINE) LoadOrdersTable,


&orders_time_start,

                                        0,


&dwThreadID[0]);

                        if (hThread[0] == NULL)
                        {

        printf("Error, failed in creating creating
thread = 0.\n");
                                exit(-1);
                        }

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

                        hThread[1] =
CreateThread(NULL,

                                        0,


(LPTHREAD_START_ROUTINE) LoadNewOrderTable,


&new_order_time_start,

                                        0,


&dwThreadID[1]);

                        if (hThread[1] == NULL)
                        {

        printf("Error, failed in creating creating
thread = 1.\n");
                                exit(-1);
                        }

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

                        hThread[2] =
CreateThread(NULL,

                                        0,


(LPTHREAD_START_ROUTINE) LoadOrderLineTable,


&order_line_time_start,

                                        0,


&dwThreadID[2]);

                        if (hThread[2] == NULL)
                        {

        printf("Error, failed in creating creating
thread = 2.\n");
                                exit(-1);
                        }

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

                        if
(CloseHandle(hThread[0]) == FALSE)
                                {

        printf("Error, failed in closing Orders
thread handle with errno: %d\n", GetLastError());
                                }

                        if
(CloseHandle(hThread[1]) == FALSE)
                                {

        printf("Error, failed in closing NewOrder
thread handle with errno: %d\n", GetLastError());
                                }

                        if
(CloseHandle(hThread[2]) == FALSE)
                                {

        printf("Error, failed in closing OrderLine
thread handle with errno: %d\n", GetLastError());
                                }
                }
        }

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

    return;
}

//=================================================
==================
//
// Function   : OrdersBufInit
//
// Clears shared buffer for ORDERS, NEWORDER, and
ORDERLINE
//
//=================================================
==================
void OrdersBufInit()
{
        int     i;
        int             j;

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

                for (j=0;j<=14;j++)
                {

        orders_buf[i].o_ol[j].ol = 0;

        orders_buf[i].o_ol[j].ol_i_id = 0;

        orders_buf[i].o_ol[j].ol_supply_w_id = 0;
```

```
          orders_buf[i].o_ol[j].ol_quantity = 0;

          orders_buf[i].o_ol[j].ol_amount = 0;

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

//================================================
====================
//
// Function   : OrdersBufLoad
//
// Fills shared buffer for ORDERS, NEWORDER, and
ORDERLINE
//
//================================================
====================
void OrdersBufLoad(short d_id, long w_id)
{

          int     cust[ORDERS_PER_DISTRICT+1];
          long    o_id;
          long      ol;

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

          GetPermutation(cust, orders_per_district);

          for
(o_id=0;o_id<orders_per_district;o_id++)
          {
                    // Generate ORDER and NEW-ORDER
data
                    orders_buf[o_id].o_d_id = d_id;
                    orders_buf[o_id].o_w_id = w_id;
                    orders_buf[o_id].o_id = o_id+1;
                    orders_buf[o_id].o_c_id =
cust[o_id+1];
                    orders_buf[o_id].o_ol_cnt =
(short)RandomNumber(5L, 15L);

                    if (o_id < first_new_order)
                    {

          orders_buf[o_id].o_carrier_id =
(short)RandomNumber(1L, 10L);

          orders_buf[o_id].o_all_local  = 1;
                    }
                    else
                    {

          orders_buf[o_id].o_carrier_id = 0;

          orders_buf[o_id].o_all_local  = 1;
                    }
```

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


          orders_buf[o_id].o_ol[ol].ol = ol+1;

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

          orders_buf[o_id].o_ol[ol].ol_supply_w_id =
w_id;

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

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

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

          FormatDate(&orders_buf[o_id].o_ol[ol].ol_de
livery_d);

                              }
                              else
                              {

          orders_buf[o_id].o_ol[ol].ol_amount =
RandomNumber(1,999999)/100.0;
                                                  // Added to
insure ol_delivery_d set properly during load

                                                  // odbc
datetime format

          strcpy(orders_buf[o_id].o_ol[ol].ol_deliver
y_d,"1899-12-31 00:00:00.000");
                              }
                    }
          }
}

//================================================
====================
//
// Function   : LoadOrdersTable
//
//================================================
====================
void LoadOrdersTable(LOADER_TIME_STRUCT
*orders_time_start)
{
          int         i;
    long        o_id;
          short     o_d_id;
```

```
          long                o_w_id;
    long        o_c_id;
    short       o_carrier_id;
    short       o_ol_cnt;
    short       o_all_local;
          char
          o_entry_d[O_ENTRY_D_LEN+1];
          RETCODE             rc;
          DBINT               rcint;

          // bind ORDER data
    i = 0;
          rc = bcp_bind(o_hdbc1, (BYTE *) &o_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);
    rc = bcp_bind(o_hdbc1, (BYTE *) &o_d_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);
          rc = bcp_bind(o_hdbc1, (BYTE *) &o_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);
    rc = bcp_bind(o_hdbc1, (BYTE *) &o_c_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);
    rc = bcp_bind(o_hdbc1, (BYTE *) &o_carrier_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);
    rc = bcp_bind(o_hdbc1, (BYTE *) &o_ol_cnt, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);
    rc = bcp_bind(o_hdbc1, (BYTE *) &o_all_local, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);
          rc = bcp_bind(o_hdbc1, (BYTE *) &o_entry_d,
0, O_ENTRY_D_LEN, NULL, 0, SQLCHARACTER, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

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

                    FormatDate(&o_entry_d);
```

```
                    // send data to server
                    rc = bcp_sendrow(o_hdbc1);
                    if (rc != SUCCEED)


          HandleErrorDBC(o_hdbc1);

                    orders_rows_loaded++;
                    CheckForCommit(o_hdbc1, o_hstmt1,
orders_rows_loaded, "orders", &orders_time_start-
>time_start);
     }

          if ((o_w_id == aptr->num_warehouses) &&
(o_d_id == 10))
          {
                    rcint = bcp_done(o_hdbc1);

                    if (rcint < 0)

          HandleErrorDBC(o_hdbc1);

                    SQLFreeStmt(o_hstmt1, SQL_DROP);
                    SQLDisconnect(o_hdbc1);
                    SQLFreeConnect(o_hdbc1);

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

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

//================================================
==================
//
// Function   : LoadNewOrderTable
//
//================================================
==================
void LoadNewOrderTable(LOADER_TIME_STRUCT
*new_order_time_start)
{
          long                i;
     long        o_id;
     short       o_d_id;
          long                o_w_id;
          RETCODE             rc;
          DBINT               rcint;

          // Bind NEW-ORDER data
          i = 0;
     rc = bcp_bind(o_hdbc2, (BYTE *) &o_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);
     rc = bcp_bind(o_hdbc2, (BYTE *) &o_d_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);
     rc = bcp_bind(o_hdbc2, (BYTE *) &o_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);

          for (i = first_new_order; i <
last_new_order; i++)
          {
                    o_id    = orders_buf[i].o_id;
                    o_d_id  = orders_buf[i].o_d_id;
                    o_w_id  = orders_buf[i].o_w_id;

                    rc = bcp_sendrow(o_hdbc2);
                    if (rc != SUCCEED)


          HandleErrorDBC(o_hdbc2);

                    new_order_rows_loaded++;

                    CheckForCommit_Big(o_hdbc2,
o_hstmt2, new_order_rows_loaded, "new_order",
&new_order_time_start->time_start);
          }

          if ((o_w_id == aptr->num_warehouses) &&
(o_d_id == 10))
          {
                    rcint = bcp_done(o_hdbc2);

                    if (rcint < 0)

          HandleErrorDBC(o_hdbc2);

                    SQLFreeStmt(o_hstmt2, SQL_DROP);
                    SQLDisconnect(o_hdbc2);
                    SQLFreeConnect(o_hdbc2);

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

//================================================
==================
//
// Function   : LoadOrderLineTable
//
//================================================
==================
void LoadOrderLineTable(LOADER_TIME_STRUCT
*order_line_time_start)
{
          long                i;
          long                j;
     long        o_id;
          short     o_d_id;
          long                o_w_id;
          double              ol;
          long      ol_i_id;
          long                ol_supply_w_id;
     short       ol_quantity;
     double      ol_amount;
     char        ol_dist_info[DIST_INFO_LEN+1];
          char
          ol_delivery_d[OL_DELIVERY_D_LEN+1];
          RETCODE             rc;
          DBINT               rcint;

          // bind ORDER-LINE data
          i = 0;
     rc = bcp_bind(o_hdbc3, (BYTE *) &o_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
     rc = bcp_bind(o_hdbc3, (BYTE *) &o_d_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
          rc = bcp_bind(o_hdbc3, (BYTE *) &o_w_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
          rc = bcp_bind(o_hdbc3, (BYTE *) &ol, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
     rc = bcp_bind(o_hdbc3, (BYTE *) &ol_i_id, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
          rc = bcp_bind(o_hdbc3, (BYTE *)
&ol_delivery_d, 0, OL_DELIVERY_D_LEN, NULL, 0,
SQLCHARACTER, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
     rc = bcp_bind(o_hdbc3, (BYTE *) &ol_amount, 0,
SQL_VARLEN_DATA, NULL, 0, SQLFLT8, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
          rc = bcp_bind(o_hdbc3, (BYTE *)
&ol_supply_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
     rc = bcp_bind(o_hdbc3, (BYTE *) &ol_quantity, 0,
SQL_VARLEN_DATA, NULL, 0, SQLINT2, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);
     rc = bcp_bind(o_hdbc3, (BYTE *) ol_dist_info, 0,
DIST_INFO_LEN, NULL, 0, 0, ++i);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

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

                    for (j=0; j <
orders_buf[i].o_ol_cnt; j++)
```

```
                    {
                              ol             =
orders_buf[i].o_ol[j].ol;
                              ol_i_id        =
orders_buf[i].o_ol[j].ol_i_id;
                              ol_supply_w_id =
orders_buf[i].o_ol[j].ol_supply_w_id;
                              ol_quantity    =
orders_buf[i].o_ol[j].ol_quantity;
                              ol_amount      =
orders_buf[i].o_ol[j].ol_amount;


          strcpy(ol_delivery_d,orders_buf[i].o_ol[j].
ol_delivery_d);


          strcpy(ol_dist_info,orders_buf[i].o_ol[j].o
l_dist_info);

                              rc =
bcp_sendrow(o_hdbc3);
                              if (rc != SUCCEED)


          HandleErrorDBC(o_hdbc3);


          order_line_rows_loaded++;


          CheckForCommit_Big(o_hdbc3, o_hstmt3,
order_line_rows_loaded, "order_line",
&order_line_time_start->time_start);
                    }
          }

          if ((o_w_id == aptr->num_warehouses) &&
(o_d_id == 10))
          {
                    rcint = bcp_done(o_hdbc3);

                    if (rcint < 0)

          HandleErrorDBC(o_hdbc3);

                    SQLFreeStmt(o_hstmt3, SQL_DROP);
                    SQLDisconnect(o_hdbc3);
                    SQLFreeConnect(o_hdbc3);

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

//================================================
==================
//
// Function   : GetPermutation
//
//================================================
==================
void GetPermutation(int perm[], int n)
{
          int i, r, t;

          for (i=1;i<=n;i++)
                    perm[i] = i;

          for (i=1;i<=n;i++)
          {
                    r = RandomNumber(i,n);
                    t = perm[i];
                    perm[i] = perm[r];
                    perm[r] = t;
          }
}

//================================================
==================
//
// Function   : CheckForCommit
//
//================================================
==================
void CheckForCommit(HDBC hdbc,

          HSTMT hstmt,

          long rows_loaded,

          char *table_name,
                                        long
*time_start)
{

    long  time_end, time_diff;

    if ( !(rows_loaded % aptr->batch) )
    {
                    time_end = (TimeNow() / MILLI);
                    time_diff = time_end -
*time_start;

                    printf("-> Loaded %ld rows into
%s in %ld sec - Total = %d (%.2f rps)\n",
                              aptr->batch,
                              table_name,
                                        time_diff,
                              rows_loaded,
                                        (float) aptr-
>batch / (time_diff ? time_diff : 1L));

                    *time_start = time_end;
    }

    return;
}

//================================================
==================
//
// Function   : CheckForCommit_Big
//
//================================================
==================
void CheckForCommit_Big(HDBC hdbc,

          HSTMT hstmt,

          double rows_loaded,

          char *table_name,
                                        long
*time_start)
{

    long  time_end, time_diff;

    if ( !(fmod(rows_loaded,aptr->batch) ) )
    {
                    time_end = (TimeNow() / MILLI);
                    time_diff = time_end -
*time_start;

                    printf("-> Loaded %ld rows into
%s in %ld sec - Total = %.0f (%.2f rps)\n",
                              aptr->batch,
                              table_name,
                                        time_diff,
                              rows_loaded,
                                        (float) aptr-
>batch / (time_diff ? time_diff : 1L));

                    *time_start = time_end;
    }

    return;
}

//================================================
==================
//
// Function   : OpenConnections
//
//================================================
==================
void OpenConnections()
{

          RETCODE         rc;

          char
          szDriverString[300];
          char
          szDriverStringOut[1024];
          SQLSMALLINT
          cbDriverStringOut;


          SQLAllocHandle(SQL_HANDLE_ENV,
SQL_NULL_HANDLE, &henv );

          SQLSetEnvAttr(henv, SQL_ATTR_ODBC_VERSION,
(void*)SQL_OV_ODBC3, 0 );
```

```
          SQLAllocHandle(SQL_HANDLE_DBC, henv ,
&i_hdbc1);
          SQLAllocHandle(SQL_HANDLE_DBC, henv ,
&w_hdbc1);
          SQLAllocHandle(SQL_HANDLE_DBC, henv ,
&c_hdbc1);
          SQLAllocHandle(SQL_HANDLE_DBC, henv ,
&c_hdbc2);
          SQLAllocHandle(SQL_HANDLE_DBC, henv ,
&o_hdbc1);
          SQLAllocHandle(SQL_HANDLE_DBC, henv ,
&o_hdbc2);
          SQLAllocHandle(SQL_HANDLE_DBC, henv ,
&o_hdbc3);

          SQLSetConnectAttr(i_hdbc1, SQL_COPT_SS_BCP,
(void *)SQL_BCP_ON, SQL_IS_INTEGER );
          SQLSetConnectAttr(w_hdbc1, SQL_COPT_SS_BCP,
(void *)SQL_BCP_ON, SQL_IS_INTEGER );
          SQLSetConnectAttr(c_hdbc1, SQL_COPT_SS_BCP,
(void *)SQL_BCP_ON, SQL_IS_INTEGER );
          SQLSetConnectAttr(c_hdbc2, SQL_COPT_SS_BCP,
(void *)SQL_BCP_ON, SQL_IS_INTEGER );
          SQLSetConnectAttr(o_hdbc1, SQL_COPT_SS_BCP,
(void *)SQL_BCP_ON, SQL_IS_INTEGER );
          SQLSetConnectAttr(o_hdbc2, SQL_COPT_SS_BCP,
(void *)SQL_BCP_ON, SQL_IS_INTEGER );
          SQLSetConnectAttr(o_hdbc3, SQL_COPT_SS_BCP,
(void *)SQL_BCP_ON, SQL_IS_INTEGER );

          // Open connections to SQL Server
          // Connection 1
          sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

            aptr->server,

            aptr->user,

            aptr->password,

            aptr->database );

          rc = SQLSetConnectOption (i_hdbc1,
SQL_PACKET_SIZE, aptr->pack_size);
          if (rc != SUCCEED)
                    HandleErrorDBC(i_hdbc1);

          rc = SQLDriverConnect ( i_hdbc1,

NULL,

          (SQLCHAR*)&szDriverString[0] ,

          SQL_NTS,

          (SQLCHAR*)&szDriverStringOut[0],

          sizeof(szDriverStringOut),

          &cbDriverStringOut,

          SQL_DRIVER_NOPROMPT );

          if ( (rc != SUCCEED) &&
                    (rc != SQL_SUCCESS_WITH_INFO) )
          {
                    HandleErrorDBC(i_hdbc1);
                    printf("TPC-C Loader
aborted!\n");
                    exit(9);
          }

          // Connection 2
          sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

            aptr->server,

            aptr->user,

            aptr->password,

            aptr->database );

          rc = SQLSetConnectOption (w_hdbc1,
SQL_PACKET_SIZE, aptr->pack_size);

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

          rc = SQLDriverConnect ( w_hdbc1,

                    NULL,

                    (SQLCHAR*)&szDriverString[0] ,

                    SQL_NTS,

                    (SQLCHAR*)&szDriverStringOut[0],

                    sizeof(szDriverStringOut),

                    &cbDriverStringOut,

                    SQL_DRIVER_NOPROMPT );

          if ( (rc != SUCCEED) &&
                    (rc != SQL_SUCCESS_WITH_INFO) )
          {
                    HandleErrorDBC(w_hdbc1);
                    printf("TPC-C Loader
aborted!\n");
                    exit(9);
          }

          // Connection 3
          sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

            aptr->server,

            aptr->user,

            aptr->password,

            aptr->database );

          rc = SQLSetConnectOption (c_hdbc1,
SQL_PACKET_SIZE, aptr->pack_size);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc1);

          rc = SQLDriverConnect ( c_hdbc1,

                    NULL,

                    (SQLCHAR*)&szDriverString[0] ,

                    SQL_NTS,

                    (SQLCHAR*)&szDriverStringOut[0],

                    sizeof(szDriverStringOut),

                    &cbDriverStringOut,

                    SQL_DRIVER_NOPROMPT );
          if ( (rc != SUCCEED) &&
                    (rc != SQL_SUCCESS_WITH_INFO) )
          {
                    HandleErrorDBC(c_hdbc1);
                    printf("TPC-C Loader
aborted!\n");
                    exit(9);
          }

          // Connection 4
          sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

            aptr->server,

            aptr->user,

            aptr->password,

            aptr->database );

          rc = SQLSetConnectOption (c_hdbc2,
SQL_PACKET_SIZE, aptr->pack_size);
          if (rc != SUCCEED)
                    HandleErrorDBC(c_hdbc2);

          rc = SQLDriverConnect ( c_hdbc2,

                    NULL,

                    (SQLCHAR*)&szDriverString[0] ,

                    SQL_NTS,

                    (SQLCHAR*)&szDriverStringOut[0],

                    sizeof(szDriverStringOut),

                    &cbDriverStringOut,

                    SQL_DRIVER_NOPROMPT );
          if ( (rc != SUCCEED) &&
```

```
                    (rc != SQL_SUCCESS_WITH_INFO) )
          {
                    HandleErrorDBC(c_hdbc2);
                    printf("TPC-C Loader
aborted!\n");
                    exit(9);
          }

          // Connection 5
          sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

            aptr->server,

            aptr->user,

            aptr->password,

            aptr->database );

          rc = SQLSetConnectOption (o_hdbc1,
SQL_PACKET_SIZE, aptr->pack_size);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc1);

          rc = SQLDriverConnect ( o_hdbc1,

                    NULL,

                    (SQLCHAR*)&szDriverString[0] ,

                    SQL_NTS,

                    (SQLCHAR*)&szDriverStringOut[0],

                    sizeof(szDriverStringOut),

                    &cbDriverStringOut,

                    SQL_DRIVER_NOPROMPT );
          if ( (rc != SUCCEED) &&
                    (rc != SQL_SUCCESS_WITH_INFO) )
          {
                    HandleErrorDBC(o_hdbc1);
                    printf("TPC-C Loader
aborted!\n");
                    exit(9);
          }

          // Connection 6
          sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

            aptr->server,

            aptr->user,

            aptr->password,

            aptr->database );

          rc = SQLSetConnectOption (o_hdbc2,
SQL_PACKET_SIZE, aptr->pack_size);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc2);

          rc = SQLDriverConnect ( o_hdbc2,

                    NULL,

                    (SQLCHAR*)&szDriverString[0] ,

                    SQL_NTS,

                    (SQLCHAR*)&szDriverStringOut[0],

                    sizeof(szDriverStringOut),

                    &cbDriverStringOut,

                    SQL_DRIVER_NOPROMPT );
          if ( (rc != SUCCEED) &&
                    (rc != SQL_SUCCESS_WITH_INFO) )
          {
                    HandleErrorDBC(o_hdbc2);
                    printf("TPC-C Loader
aborted!\n");
                    exit(9);
          }

          // Connection 7
          sprintf( szDriverString , "DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=%s" ,

            aptr->server,

            aptr->user,

            aptr->password,

            aptr->database );

          rc = SQLSetConnectOption (o_hdbc3,
SQL_PACKET_SIZE, aptr->pack_size);
          if (rc != SUCCEED)
                    HandleErrorDBC(o_hdbc3);

          rc = SQLDriverConnect ( o_hdbc3,

                    NULL,

                    (SQLCHAR*)&szDriverString[0] ,

                    SQL_NTS,

                    (SQLCHAR*)&szDriverStringOut[0],

                    sizeof(szDriverStringOut),

                    &cbDriverStringOut,

                    SQL_DRIVER_NOPROMPT );
          if ( (rc != SUCCEED) &&
                    (rc != SQL_SUCCESS_WITH_INFO) )
          {
                    HandleErrorDBC(o_hdbc3);
                    printf("TPC-C Loader
aborted!\n");
                    exit(9);
          }
}

//================================================
==================
//
// Function name: BuildIndex
//
//================================================
==================
void BuildIndex(char          *index_script)
{
          char      cmd[256];

          printf("Starting index creation:
%s\n",index_script);

          sprintf(cmd, "osql -S%s -U%s -P%s -e -
i%s\\%s.sql > %s%s.log",
                                        aptr->server,
                                        aptr->user,
                                        aptr-
>password,
                                        aptr-
>index_script_path,
                                        index_script,
                                        aptr-
>log_path,

          index_script);

          system(cmd);

          printf("Finished index creation:
%s\n",index_script);
}

//================================================
==================
//
// Function name: HandleErrorDBC
//
//================================================
==================
void HandleErrorDBC (SQLHDBC  hdbc1)
{

          SQLCHAR             SqlState[6],
Msg[SQL_MAX_MESSAGE_LENGTH];
          SQLLEN              NativeError;
          SQLSMALLINT i, MsgLen;
          SQLRETURN   rc2;
          char                timebuf[128];
          char                datebuf[128];
          char                err_log_path[256];
          FILE                *fp1;

          i = 1;
          while (( rc2 = SQLGetDiagRec(SQL_HANDLE_DBC
, hdbc1, i, SqlState , &NativeError,
```

```
                                        Msg,
sizeof(Msg) , &MsgLen )) != SQL_NO_DATA )
        {
                sprintf( szLastError , "%s" ,
Msg );

                _strtime(timebuf);
                _strdate(datebuf);

                printf( "[%s : %s]
%s\n==>SQLState: %s\n" , datebuf, timebuf,
szLastError, SqlState);

                strcpy(err_log_path,aptr-
>log_path);

        strcat(err_log_path,"tpccldr.err");
                fp1 = fopen(err_log_path,"a+");
                if (fp1 == NULL)
                        printf("ERROR:  Unable
to open errorlog file.\n");
                else
                {
                        fprintf(fp1, "[%s : %s]
%s\nSQLState: %s\n" , datebuf, timebuf, szLastError,
SqlState);
                        fclose(fp1);
                }

                i++;
        }
}

//=================================================
==================
//
// Function   : HandleErrorSTMT
//
//=================================================
==================
void HandleErrorSTMT (HSTMT  hstmt1)
{
        SQLCHAR             SqlState[6],
Msg[SQL_MAX_MESSAGE_LENGTH];
        SQLLEN              NativeError;
        SQLSMALLINT i, MsgLen;
        SQLRETURN   rc2;
        char                timebuf[128];
        char                datebuf[128];
        char                err_log_path[256];
        FILE                *fp1;

        i = 1;
        while (( rc2 =
SQLGetDiagRec(SQL_HANDLE_STMT , hstmt1, i, SqlState ,
&NativeError,
                                        Msg,
sizeof(Msg) , &MsgLen )) != SQL_NO_DATA )
        {
                if (total_db_errors >=
MAX_SQL_ERRORS)
                {
                        printf(">>>>> Maximum
SQL errors of %d exceeded.  Terminating
TPCCLDR.<<<<<\n",total_db_errors);
                        exit(9);
                }
                total_db_errors++;

                sprintf( szLastError , "%s" ,
Msg );

                _strtime(timebuf);
                _strdate(datebuf);

                printf( "[%s : %s] %s\nSQLState:
%s\n" , datebuf, timebuf, szLastError, SqlState);

                strcpy(err_log_path,aptr-
>log_path);

        strcat(err_log_path,"tpccldr.err");
                fp1 = fopen(err_log_path,"a+");
                if (fp1 == NULL)
                        printf("ERROR:  Unable
to open errorlog file.\n");
                else
                {
                        fprintf(fp1, "[%s : %s]
%s\nSQLState: %s\n" , datebuf, timebuf, szLastError,
SqlState);
                        fclose(fp1);
                }

                i++;
        }
}

//=================================================
==================
//
// Function   : FormatDate
//
//=================================================
==================
void FormatDate ( char* szTimeCOutput )
{
        struct tm when;
        time_t now;

        time( &now );
        when  = *localtime( &now );

        mktime( &when );

        // odbc datetime format
        strftime( szTimeCOutput , 30 , "%Y-%m-%d
%H:%M:%S.000", &when );

        return;
}
```

tpcc_neworder_new.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   TPCC_NEWORDER_NEW.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          This acid stored procedure implements the
neworder  --
--          transaction.  It outputs timestamps at
the          --
--          beginning of the transaction, before the
commit     --
--          delay, and after the commit.
--
--
--
-------------------------------------------------------
-------------
SET QUOTED_IDENTIFIER OFF
GO
SET ANSI_NULLS OFF
GO

USE tpcc
GO

IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_neworder_new' )
    DROP PROCEDURE tpcc_neworder_new
GO

-- neworder_new v2.5 6/23/05 PeterCa
-- 1q stock/order_line/client.  upd district & ins
neworder.
--  cust/warehouse select together, ins order
separate
--  uses rownumber to distinct w any transform
--  uses in-memory sort for distinct on iid,wid
--  uses charindex
-- will rollback if (@i_idX,@s_w_idX pairs not
unique) OR (@i_idX not unique).

CREATE PROCEDURE    tpcc_neworder_new
                        @w_id           int,
                        @d_id           tinyint,
                        @c_id           int,
                        @o_ol_cnt       tinyint,
                        @o_all_local    tinyint,
                        @i_id1  int = 0, @s_w_id1
int = 0, @ol_qty1  smallint = 0,
                        @i_id2  int = 0, @s_w_id2
int = 0, @ol_qty2  smallint = 0,
```

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



AS
BEGIN
DECLARE @o_id           int,
        @d_tax          smallmoney,
        @o_entry_d      datetime,
        @commit_flag    tinyint

BEGIN TRANSACTION n
    --  get district tax and next availible order id
and update
    --  insert corresponding row into new-order table
    --  plus initialize local variables

    UPDATE  district
    SET     @d_tax          = d_tax,
            @o_id           = d_next_o_id,
            d_next_o_id     = d_next_o_id + 1,
            @o_entry_d      = GETDATE(),
            @commit_flag    = 1
    OUTPUT  deleted.d_next_o_id,
            @d_id,
            @w_id
    INTO    new_order
    WHERE   d_w_id          = @w_id AND
            d_id            = @d_id

    -- update stock from stock join (item join
(params))
    -- output to orderline, output to client
    -- NOTE: @@rowcount != @ol_o_cnt
    --    if (@i_idX,@s_w_idX pairs not unique) OR
(@i_idX not unique).

    UPDATE  stock
    SET     s_ytd           = s_ytd + info.ol_qty,
            s_quantity      = s_quantity -
info.ol_qty +
                              CASE WHEN (s_quantity -
info.ol_qty < 10) THEN 91 ELSE 0 END,
            s_order_cnt     = s_order_cnt + 1,
            s_remote_cnt    = s_remote_cnt +
                                                  CASE
WHEN (info.w_id = @w_id) THEN 0 ELSE 1 END

    OUTPUT  @o_id,
            @d_id,
            @w_id,
            info.lino,
            info.i_id,
            "dec 31, 1899",
                  info.i_price * info.ol_qty,
                  info.w_id,
            info.ol_qty,
            CASE    @d_id   WHEN 1  THEN
inserted.s_dist_01
                            WHEN 2  THEN
inserted.s_dist_02
                            WHEN 3  THEN
inserted.s_dist_03
                            WHEN 4  THEN
inserted.s_dist_04
                            WHEN 5  THEN
inserted.s_dist_05
                            WHEN 6  THEN
inserted.s_dist_06
                            WHEN 7  THEN
inserted.s_dist_07
                            WHEN 8  THEN
inserted.s_dist_08
                            WHEN 9  THEN
inserted.s_dist_09
                            WHEN 10 THEN
inserted.s_dist_10
            END
    INTO    order_line

    OUTPUT  info.i_name,inserted.s_quantity,
            CASE WHEN
((charindex("ORIGINAL",info.i_data) > 0) AND

(charindex("ORIGINAL",inserted.s_data) > 0) )
                  THEN "B" ELSE "G" END,
            info.i_price,
            info.i_price*info.ol_qty
    FROM    stock INNER JOIN
            (SELECT iid,
                    wid,
                          lino,
                    ol_qty,
                    i_price,
                    i_name,
                    i_data
             FROM   (SELECT iid,
                            wid,
                            lino,
                            qty,
                                        row_number()
OVER (PARTITION BY iid,wid ORDER BY iid,wid)
                           FROM   (SELECT
@i_id1,@s_w_id1,1,@ol_qty1      UNION ALL
                            SELECT
@i_id2,@s_w_id2,2,@ol_qty2      UNION ALL
                            SELECT
@i_id3,@s_w_id3,3,@ol_qty3      UNION ALL
                            SELECT
@i_id4,@s_w_id4,4,@ol_qty4      UNION ALL
                            SELECT
@i_id5,@s_w_id5,5,@ol_qty5      UNION ALL
                            SELECT
@i_id6,@s_w_id6,6,@ol_qty6      UNION ALL
                            SELECT
@i_id7,@s_w_id7,7,@ol_qty7      UNION ALL
                            SELECT
@i_id8,@s_w_id8,8,@ol_qty8      UNION ALL
                            SELECT
@i_id9,@s_w_id9,9,@ol_qty9      UNION ALL
                            SELECT
@i_id10,@s_w_id10,10,@ol_qty10  UNION ALL
                            SELECT
@i_id11,@s_w_id11,11,@ol_qty11  UNION ALL
                            SELECT
@i_id12,@s_w_id12,12,@ol_qty12  UNION ALL
                            SELECT
@i_id13,@s_w_id13,13,@ol_qty13  UNION ALL
                            SELECT
@i_id14,@s_w_id14,14,@ol_qty14  UNION ALL
                            SELECT
@i_id15,@s_w_id15,15,@ol_qty15) AS
uo1(iid,wid,lino,qty)
                                 ) AS
o1(iid,wid,lino,ol_qty,rownum)
                           INNER JOIN
                           item (repeatableread) ON
i_id   = iid  AND -- filters out invalid items

rownum = 1
                     ) AS
info(i_id,w_id,lino,ol_qty,i_price,i_name,i_data)
                ON  s_i_id  = info.i_id AND
                    s_w_id  = info.w_id

    IF (@@rowcount <> @o_ol_cnt)  -- must have an
invalid item
        SELECT  @commit_flag = 0  -- 2.4.2.3 requires
rest to proceed

    --  insert fresh row into orders table
    INSERT INTO orders VALUES   (   @o_id,
                                    @d_id,
                                    @w_id,
                                    @c_id,
                                    0,
                                    @o_ol_cnt,
                                    @o_all_local,
                                    @o_entry_d)


    --  get customer last name, discount, and credit
rating
    --  get warehouse tax
```

```
    --  return order_data to client
    SELECT  w_tax,
            @d_tax,
            @o_id,
            c_last,
            c_discount,
            c_credit,
            @o_entry_d,
            @commit_flag
    FROM    warehouse(repeatableread),
            customer(repeatableread)
    WHERE   w_id    = @w_id AND
            c_id    = @c_id AND
            c_w_id  = @w_id AND
            c_d_id  = @d_id

          -- @@rowcount checks that previous select
found a valid customer
    IF (@@rowcount = 0)
    BEGIN
                    RAISERROR( 'Invalid Customer ID',
11, 1 )
        ROLLBACK TRANSACTION n
    END
         ELSE IF (@commit_flag = 1)
        COMMIT TRANSACTION n
    ELSE    -- all that work for nothing.
        ROLLBACK TRANSACTION  n

END
GO
```

VerifyTPCCLoad.sql

```
-------------------------------------------------------
-------------
--
--
--  File:   VerifyTPCCLoad.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
-------------------------------------------------------
-------------
SET NOCOUNT ON
PRINT   ' '
SELECT  CONVERT(CHAR(30), GETDATE(), 21)
PRINT   ' '

USE       tpcc
GO

IF  EXISTS  (SELECT name
             FROM   sysobjects
             WHERE  name    = 'TPCC_INFO' AND
                      type    = 'U')
    DROP TABLE TPCC_INFO
GO
PRINT   'WAREHOUSE TABLE'
SELECT  count_big(*)
FROM    warehouse
GO

PRINT   'DISTRICT TABLE = (10 * No of warehouses)'
SELECT  count_big(*)
FROM    district
GO

PRINT   'ITEM TABLE = 100,000'
SELECT  count_big(*)
FROM    item
GO

PRINT   'CUSTOMER TABLE = (30,000 * No of
warehouses)'
SELECT  count_big(*)
FROM    customer
GO

PRINT   'ORDERS TABLE = (30,000 * No of warehouses)'
SELECT  count_big(*)
FROM    orders
GO

PRINT   'HISTORY TABLE = (30,000 * No of warehouses)'
SELECT  count_big(*)
FROM    history
GO

PRINT   'STOCK TABLE = (100,000 * No of warehouses)'
SELECT  count_big(*)
FROM    stock
GO

PRINT   'ORDER_LINE TABLE = (300,000 * No of
warehouses + some change)'
SELECT  count_big(*)
FROM    order_line
GO

PRINT   'NEW_ORDER TABLE = (9000 * No of warehouses)'
SELECT  count_big(*)
FROM    new_order
GO

CREATE TABLE TPCC_INFO
    (   INFO_DATE                   datetime,
        NUM_WAREHOUSE               bigint,
        WAREHOUSE_TARGET            bigint,
        NUM_DISTRICT                bigint,
        DISTRICT_TARGET             bigint,
        NUM_ITEM                    bigint,
        ITEM_TARGET                 bigint,
        NUM_CUSTOMER                bigint,
        CUSTOMER_TARGET             bigint,
        NUM_ORDERS                  bigint,
        ORDERS_TARGET               bigint,
        ORDERS_TARGET_LOW           bigint,
        ORDERS_TARGET_HIGH          bigint,
        NUM_ORDER_LINE              bigint,
        ORDER_LINE_TARGET           bigint,
        ORDER_LINE_TARGET_LOW       bigint,
        ORDER_LINE_TARGET_HIGH      bigint,
        NUM_NEW_ORDER               bigint,
        NEW_ORDER_TARGET            bigint,
        NEW_ORDER_TARGET_LOW        bigint,
        NEW_ORDER_TARGET_HIGH       bigint,
        NUM_HISTORY                 bigint,
        HISTORY_TARGET              bigint,
        NUM_STOCK                   bigint,
        STOCK_TARGET                bigint)
GO

DECLARE @NUM_WAREHOUSE              bigint,
        @WAREHOUSE_TARGET           bigint,
        @NUM_DISTRICT               bigint,
        @DISTRICT_TARGET            bigint,
        @NUM_ITEM                   bigint,
        @ITEM_TARGET                bigint,
        @NUM_CUSTOMER               bigint,
        @CUSTOMER_TARGET            bigint,
        @NUM_ORDERS                 bigint,
        @ORDERS_TARGET              bigint,
        @ORDERS_TARGET_LOW          bigint,
        @ORDERS_TARGET_HIGH         bigint,
        @NUM_ORDER_LINE             bigint,
        @ORDER_LINE_TARGET          bigint,
        @ORDER_LINE_TARGET_LOW      bigint,
        @ORDER_LINE_TARGET_HIGH     bigint,
        @NUM_NEW_ORDER              bigint,
        @NEW_ORDER_TARGET           bigint,
        @NEW_ORDER_TARGET_LOW       bigint,
        @NEW_ORDER_TARGET_HIGH      bigint,
        @NUM_HISTORY                bigint,
        @HISTORY_TARGET             bigint,
        @NUM_STOCK                  bigint,
        @STOCK_TARGET               bigint

-- set the local variables prior to inserting them
into the TPCC_INFO table
SELECT  @NUM_WAREHOUSE      = COUNT_BIG(*)
FROM    warehouse

SELECT  @NUM_DISTRICT       = COUNT_BIG(*)
FROM    district

SELECT  @NUM_ITEM           = COUNT_BIG(*)
FROM    item

SELECT  @NUM_CUSTOMER       = COUNT_BIG(*)
FROM    customer

SELECT  @NUM_ORDERS         = COUNT_BIG(*)
FROM    orders

SELECT  @NUM_ORDER_LINE     = COUNT_BIG(*)
FROM    order_line

SELECT  @NUM_NEW_ORDER      = COUNT_BIG(*)
FROM    new_order
```

```
SELECT  @NUM_HISTORY        = COUNT_BIG(*)
FROM    history

SELECT  @NUM_STOCK          = COUNT_BIG(*)
FROM    stock

--- now calculate and set the target values
SELECT  @WAREHOUSE_TARGET       = @NUM_WAREHOUSE,
        @DISTRICT_TARGET        = @NUM_WAREHOUSE *
10,
        @ITEM_TARGET            = 100000,
        @CUSTOMER_TARGET        = @NUM_WAREHOUSE *
30000,
        @ORDERS_TARGET          = @NUM_WAREHOUSE *
30000,
        @ORDERS_TARGET_LOW      = @ORDERS_TARGET -
FLOOR(@ORDERS_TARGET * .01),
        @ORDERS_TARGET_HIGH     = @ORDERS_TARGET +
FLOOR(@ORDERS_TARGET * .01),
        @ORDER_LINE_TARGET      = @NUM_WAREHOUSE *
300000,
        @ORDER_LINE_TARGET_LOW  = @ORDER_LINE_TARGET
- FLOOR(@ORDER_LINE_TARGET * .01),
        @ORDER_LINE_TARGET_HIGH = @ORDER_LINE_TARGET
+ FLOOR(@ORDER_LINE_TARGET * .01),
        @NEW_ORDER_TARGET       = @NUM_WAREHOUSE *
9000,
        @NEW_ORDER_TARGET_LOW   = @NEW_ORDER_TARGET -
FLOOR(@NEW_ORDER_TARGET * .01),
        @NEW_ORDER_TARGET_HIGH  = @NEW_ORDER_TARGET +
FLOOR(@NEW_ORDER_TARGET * .01),
        @HISTORY_TARGET         = @NUM_WAREHOUSE *
30000,
        @STOCK_TARGET           = @NUM_WAREHOUSE *
100000

--- insert the values into TPCC_INFO
INSERT INTO TPCC_INFO VALUES    (GETDATE(),
                                 @NUM_WAREHOUSE,
                                 @WAREHOUSE_TARGET,
                                 @NUM_DISTRICT,
                                 @DISTRICT_TARGET,
                                 @NUM_ITEM,
                                 @ITEM_TARGET,
                                 @NUM_CUSTOMER,
                                 @CUSTOMER_TARGET,
                                 @NUM_ORDERS,
                                 @ORDERS_TARGET,
                                 @ORDERS_TARGET_LOW,
                                 @ORDERS_TARGET_HIGH,
                                 @NUM_ORDER_LINE,
                                 @ORDER_LINE_TARGET,

@ORDER_LINE_TARGET_LOW,

@ORDER_LINE_TARGET_HIGH,
                                 @NUM_NEW_ORDER,
                                 @NEW_ORDER_TARGET,

@NEW_ORDER_TARGET_LOW,

@NEW_ORDER_TARGET_HIGH,
                                 @NUM_HISTORY,
                                 @HISTORY_TARGET,
                                 @NUM_STOCK,
                                 @STOCK_TARGET)
GO

--- output the row counts from the build
PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|    WAREHOUSE TABLE      |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_WAREHOUSE                   AS
'Warehouse Rows',
        WAREHOUSE_TARGET                AS
          'Warehouse Target',
        CASE WHEN (NUM_WAREHOUSE = WAREHOUSE_TARGET)
             THEN 'OK!'
             ELSE 'ERROR!!!'
        END                             AS
'Warehouse Message'
FROM    TPCC_INFO
GO

PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|    DISTRICT TABLE       |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_DISTRICT                    AS  'District
Rows',
        DISTRICT_TARGET                 AS
          'District Target',
        CASE WHEN (NUM_DISTRICT = DISTRICT_TARGET)
             THEN 'OK!'
             ELSE 'ERROR!!!'
        END                             AS  'District
Message'
FROM    TPCC_INFO
GO

PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|      ITEM TABLE         |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_ITEM                        AS  'Item
Rows',
        ITEM_TARGET                     AS
          'Item Target',
        CASE WHEN (NUM_ITEM = ITEM_TARGET)
             THEN 'OK!'
             ELSE 'ERROR!!!'
        END                             AS  'Item
Message'
FROM    TPCC_INFO
GO

PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|    CUSTOMER TABLE       |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_CUSTOMER                    AS  'Customer
Rows',
        CUSTOMER_TARGET                 AS
          'Customer Target',
        CASE WHEN (NUM_CUSTOMER = CUSTOMER_TARGET)
             THEN 'OK!'
             ELSE 'ERROR!!!'
        END                             AS  'Customer
Message'
FROM    TPCC_INFO
GO

PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|     ORDERS TABLE        |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_ORDERS                      AS  'Orders
Rows',
        ORDERS_TARGET                   AS
          'Orders Target',
        CASE WHEN (NUM_ORDERS = ORDERS_TARGET)
             THEN 'OK!'
             WHEN (NUM_ORDERS BETWEEN
ORDERS_TARGET_LOW AND ORDERS_TARGET_HIGH)
             THEN 'OK! (within 1%)'
             ELSE 'ERROR!!!'
        END                             AS  'Orders
Message'
FROM    TPCC_INFO
GO

PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|   ORDER LINE TABLE      |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_ORDER_LINE                  AS  'Order
Line Rows',
        ORDER_LINE_TARGET               AS
          'Order Line Target',
        CASE WHEN (NUM_ORDER_LINE =
ORDER_LINE_TARGET)
             THEN 'OK!'
             WHEN (NUM_ORDER_LINE BETWEEN
ORDER_LINE_TARGET_LOW AND ORDER_LINE_TARGET_HIGH)
             THEN 'OK! (within 1%)'
             ELSE 'ERROR!!!'
        END                             AS  'Order
Line Message'
FROM    TPCC_INFO
GO
```

```
PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|    NEW ORDER TABLE      |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_NEW_ORDER                   AS  'New
Order Rows',
        NEW_ORDER_TARGET                AS
          'New Order Target',
        CASE WHEN (NUM_NEW_ORDER = NEW_ORDER_TARGET)
             THEN 'OK!'
             WHEN (NUM_NEW_ORDER BETWEEN
NEW_ORDER_TARGET_LOW AND NEW_ORDER_TARGET_HIGH)
             THEN 'OK! (within 1%)'
             ELSE 'ERROR!!!'
        END                             AS  'New
Order Message'
FROM    TPCC_INFO
GO

PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|     HISTORY TABLE       |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_HISTORY                     AS  'History
Rows',
        HISTORY_TARGET                  AS
          'History Target',
        CASE WHEN (NUM_HISTORY = HISTORY_TARGET)
             THEN 'OK!'
             ELSE 'ERROR!!!'
        END                             AS  'History
Message'
FROM    TPCC_INFO
GO

PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|      STOCK TABLE        |'
PRINT   '---------------------------'
SELECT  TOP 1
        CONVERT(CHAR(30),INFO_DATE,21)  AS  'Date',
        NUM_STOCK                       AS  'Stock
Rows',
        STOCK_TARGET                    AS
          'Stock Target',
        CASE WHEN (NUM_STOCK = STOCK_TARGET)
             THEN 'OK!'
             ELSE 'ERROR!!!'
        END                             AS  'Stock
Message'
FROM    TPCC_INFO
GO

------------------
-- Check Indexes
------------------
USE tpcc
GO

PRINT   ''
PRINT   ''
PRINT   '---------------------------'
PRINT   '|      TPC-C INDEXES       |'
PRINT   '---------------------------'
EXEC    sp_helpindex    warehouse
EXEC    sp_helpindex    district
EXEC    sp_helpindex    item
EXEC    sp_helpindex    customer
EXEC    sp_helpindex    orders
EXEC    sp_helpindex    order_line
EXEC    sp_helpindex    new_order
EXEC    sp_helpindex    history
EXEC    sp_helpindex    stock
GO
```

version.sql

```
------------------------------------------------------
-------------
--
--
--  File:   VERSION.SQL
--
--          Microsoft TPC-C Benchmark Kit Ver. 4.68
--
--          Copyright Microsoft, 2006
--
--
--
--          Returns version level of TPC-C stored
procs         --
--
--
--          Always update the return value of this
proc for     --
--          any interface changes or 'must have' bug
fixes.     --
--
--
--          The value returned by this SP defines the
--
--          'interface level', which must match
between the      --
--          stored procs and the client code.  The
--
--          interface level may be down rev from the
--
--          current kit.  This indicates that the
interface     --
--          hasn't changed since that version.
--
--
--
--          Interface Level:    4.20.000
--
--
--
------------------------------------------------------
-------------
USE tpcc
GO

IF EXISTS ( SELECT name FROM sysobjects WHERE name =
'tpcc_version' )
             DROP PROCEDURE tpcc_version
GO

CREATE PROCEDURE    tpcc_version
AS
DECLARE @version    char(8)

BEGIN
    SELECT  @version    = '4.20.000'

    SELECT  @version    AS 'Version'
END
GO
```

Appendix C:

Tunable Parameters

dl380g7-83968wh-profile.txt

```
Profile:  dl380g7-83968wh-profile
File Path:          C:\scripts\profile_gen\dl380g7-
83968wh-profile.xml
Version:  5

         Number of Engines: 128

                  Name: d1
                  Description:
                  Directory: c:\d2.log
                  Machine: n90
                  Parameter Set: 2.2
                  Index: 100000000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER53164609
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 0
                  Additional Options:

                  Name: d2
                  Description:
                  Directory: c:\d1.log
                  Machine: n90
                  Parameter Set: 2.2
                  Index: 100300000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER44265281
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 1
                  Additional Options:

                  Name: d3
                  Description:
                  Directory: c:\d3.log
                  Machine: n90
                  Parameter Set: 2.2
                  Index: 100600000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER3439676359
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 4
                  Additional Options:

                  Name: d4
                  Description:
                  Directory: c:\d4.log
                  Machine: n90
                  Parameter Set: 2.2
                  Index: 100900000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER4439706187
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 5
                  Additional Options:

                  Name: d5
                  Description:
                  Directory: c:\d5.log
                  Machine: n91
                  Parameter Set: 2.2
                  Index: 101200000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER5346413218
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 0
                  Additional Options:

                  Name: d6
                  Description:
                  Directory: c:\d6.log
                  Machine: n91
                  Parameter Set: 2.2
                  Index: 101500000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER62226046
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 1
                  Additional Options:

                  Name: d7
                  Description:
                  Directory: c:\d7.log
                  Machine: n91
                  Parameter Set: 2.2
                  Index: 101800000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER72289718
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 4
                  Additional Options:

                  Name: d8
                  Description:
                  Directory: c:\d8.log
                  Machine: n91
                  Parameter Set: 2.2
                  Index: 102100000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER82325578
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 5
                  Additional Options:

                  Name: d9
                  Description:
                  Directory: c:\d9.log
                  Machine: n92
                  Parameter Set: 2.2
                  Index: 102400000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER92360187
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 0
                  Additional Options:

                  Name: d10
                  Description:
                  Directory: c:\d10.log
                  Machine: n92
                  Parameter Set: 2.2
                  Index: 102700000
                  Seed: 4678
                  Configured Users: 6560
                  Pipe Name: DRIVER102399796
                  Connect Rate: 10
                  Start Rate: 0
                  Max. Concurrency: 0
                  Concurrency Rate: 0
                  CLIENT_NURAND: 25
                  CPU: 1
                  Additional Options:

                  Name: d11
                  Description:
                  Directory: c:\d11.log
                  Machine: n92
```

```
Parameter Set: 2.2
Index: 103000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER1122682203
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d12
Description:
Directory: c:\d12.log
Machine: n92
Parameter Set: 2.2
Index: 103300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER1222731546
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d13
Description:
Directory: c:\d13.log
Machine: n93
Parameter Set: 2.2
Index: 103600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER13-1439076421
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d14
Description:
Directory: c:\d14.log
Machine: n93
Parameter Set: 2.2
Index: 103900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER14-1438943656
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d15
Description:
Directory: c:\d15.log
Machine: n93
Parameter Set: 2.2
Index: 104200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER15-1438852265
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d16
Description:
Directory: c:\d16.log
Machine: n93
Parameter Set: 2.2
Index: 104500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER16-1438790906
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d17
Description:
Directory: c:\d17.log
Machine: n94
Parameter Set: 2.2
Index: 104800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER17-57150250
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d18
Description:
Directory: c:\d18.log
Machine: n94
Parameter Set: 2.2
Index: 105100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER18-57076468
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d19
Description:
Directory: c:\d19.log
Machine: n94
Parameter Set: 2.2
Index: 105400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER19-57030562
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d20
Description:
Directory: c:\d20.log
Machine: n94
Parameter Set: 2.2
Index: 105700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER20-56992625
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d21
Description:
Directory: c:\d21.log
Machine: n95
Parameter Set: 2.2
Index: 106000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER2191781
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d22
Description:
Directory: c:\d22.log
Machine: n95
Parameter Set: 2.2
Index: 106300000
```

```
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER221814250
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d23
Description:
Directory: c:\d23.log
Machine: n95
Parameter Set: 2.2
Index: 106600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER231877968
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d24
Description:
Directory: c:\d24.log
Machine: n95
Parameter Set: 2.2
Index: 106900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER242206343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d25
Description:
Directory: c:\d25.log
Machine: n96
Parameter Set: 2.2
Index: 107200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER252251500
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d26
Description:
Directory: c:\d26.log
Machine: n96
Parameter Set: 2.2
Index: 107500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER262289250
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d27
Description:
Directory: c:\d27.log
Machine: n96
Parameter Set: 2.2
Index: 107800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER272340437
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d28
Description:
Directory: c:\d28.log
Machine: n96
Parameter Set: 2.2
Index: 108100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER282382234
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d29
Description:
Directory: c:\d29.log
Machine: n97
Parameter Set: 2.2
Index: 108400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER292416328
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d30
Description:
Directory: c:\d30.log
Machine: n97
Parameter Set: 2.2
Index: 108700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER302463687
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d31
Description:
Directory: c:\d31.log
Machine: n97
Parameter Set: 2.2
Index: 109000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER3155814328
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d32
Description:
Directory: c:\d32.log
Machine: n97
Parameter Set: 2.2
Index: 109300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER3255892765
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d33
Description:
Directory: c:\d33.log
Machine: n98
Parameter Set: 2.2
Index: 109600000
Seed: 4678
Configured Users: 6560
```

```
Pipe Name: DRIVER3355948500
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d34
Description:
Directory: c:\d34.log
Machine: n98
Parameter Set: 2.2
Index: 109900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER3455990593
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d35
Description:
Directory: c:\d35.log
Machine: n98
Parameter Set: 2.2
Index: 110200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER3556027390
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d36
Description:
Directory: c:\d36.log
Machine: n98
Parameter Set: 2.2
Index: 110500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER3656077062
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d37
Description:
Directory: c:\d37.log
Machine: n99
Parameter Set: 2.2
Index: 110800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER37766536203
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d38
Description:
Directory: c:\d38.log
Machine: n99
Parameter Set: 2.2
Index: 111100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER38766654375
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d39
Description:
Directory: c:\d39.log
Machine: n99
Parameter Set: 2.2
Index: 111400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER39766760968
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d40
Description:
Directory: c:\d40.log
Machine: n99
Parameter Set: 2.2
Index: 111700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER40766820328
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d41
Description:
Directory: c:\d38.log
Machine: n100
Parameter Set: 2.2
Index: 112000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER41766909890
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d42
Description:
Directory: c:\d42.log
Machine: n100
Parameter Set: 2.2
Index: 112300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER42766941343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d43
Description:
Directory: c:\d43.log
Machine: n100
Parameter Set: 2.2
Index: 112600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER43766990906
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d44
Description:
Directory: c:\d44.log
Machine: n100
Parameter Set: 2.2
Index: 112900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER44767023437
Connect Rate: 10
```

```
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d45
Description:
Directory: c:\d45.log
Machine: n101
Parameter Set: 2.2
Index: 113200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER45767085000
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d46
Description:
Directory: c:\d46.log
Machine: n101
Parameter Set: 2.2
Index: 113500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER46767120687
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d47
Description:
Directory: c:\d47.log
Machine: n101
Parameter Set: 2.2
Index: 113800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER47767168296
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d48
Description:
Directory: c:\d48.log
Machine: n101
Parameter Set: 2.2
Index: 114100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER48767212015
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d49
Description:
Directory: c:\d49.log
Machine: n102
Parameter Set: 2.2
Index: 114400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER49778610406
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d50
Description:
Directory: c:\d50.log
Machine: n102
Parameter Set: 2.2
Index: 114700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER50778666593
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d51
Description:
Directory: c:\d51.log
Machine: n102
Parameter Set: 2.2
Index: 115000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER51778705953
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d52
Description:
Directory: c:\d52.log
Machine: n102
Parameter Set: 2.2
Index: 115300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER52778774546
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d53
Description:
Directory: c:\d53.log
Machine: n103
Parameter Set: 2.2
Index: 115600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER53778801906
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d54
Description:
Directory: c:\d54.log
Machine: n103
Parameter Set: 2.2
Index: 115900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER54778828968
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d55
Description:
Directory: c:\d55.log
Machine: n103
Parameter Set: 2.2
Index: 116200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER55778888203
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
```

```
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d56
Description:
Directory: c:\d56.log
Machine: n103
Parameter Set: 2.2
Index: 116500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER56778926656
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d57
Description:
Directory: c:\d57.log
Machine: n104
Parameter Set: 2.2
Index: 116800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER57778954765
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d58
Description:
Directory: c:\d58.log
Machine: n104
Parameter Set: 2.2
Index: 117100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER58778987609
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d59
Description:
Directory: c:\d59.log
Machine: n104
Parameter Set: 2.2
Index: 117400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER59779021390
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d60
Description:
Directory: c:\d60.log
Machine: n104
Parameter Set: 2.2
Index: 117700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER60779145406
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d61
Description:
Directory: c:\d61.log
Machine: n105
Parameter Set: 2.2
Index: 118000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER613345406
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d62
Description:
Directory: c:\d62.log
Machine: n105
Parameter Set: 2.2
Index: 118300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER623453375
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d63
Description:
Directory: c:\d63.log
Machine: n105
Parameter Set: 2.2
Index: 118600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER633501687
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d64
Description:
Directory: c:\d64.log
Machine: n105
Parameter Set: 2.2
Index: 118900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER643542156
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d65
Description:
Directory: c:\d65.log
Machine: n106
Parameter Set: 2.2
Index: 119200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER653612937
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d66
Description:
Directory: c:\d66.log
Machine: n106
Parameter Set: 2.2
Index: 119500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER663655140
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
```

```
CPU: 1
Additional Options:

Name: d67
Description:
Directory: c:\d67.log
Machine: n106
Parameter Set: 2.2
Index: 119800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER673761906
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d68
Description:
Directory: c:\d68.log
Machine: n106
Parameter Set: 2.2
Index: 120100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER683819031
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d69
Description:
Directory: c:\d69.log
Machine: n107
Parameter Set: 2.2
Index: 120400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER693865343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d70
Description:
Directory: c:\d70.log
Machine: n107
Parameter Set: 2.2
Index: 120700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER703910750
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d71
Description:
Directory: c:\d71.log
Machine: n107
Parameter Set: 2.2
Index: 121000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER713949343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d72
Description:
Directory: c:\d72.log
Machine: n107
Parameter Set: 2.2
Index: 121300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER723985750
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d73
Description:
Directory: c:\d73.log
Machine: n108
Parameter Set: 2.2
Index: 121600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER732742140
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d74
Description:
Directory: c:\d74.log
Machine: n108
Parameter Set: 2.2
Index: 121900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER742768187
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d75
Description:
Directory: c:\d75.log
Machine: n108
Parameter Set: 2.2
Index: 122200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER752779937
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d76
Description:
Directory: c:\d76.log
Machine: n108
Parameter Set: 2.2
Index: 122500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER762790703
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d77
Description:
Directory: c:\d77.log
Machine: n109
Parameter Set: 2.2
Index: 122800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER772802046
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:
```

```
Name: d78
Description:
Directory: c:\d78.log
Machine: n109
Parameter Set: 2.2
Index: 123100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER782810718
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d79
Description:
Directory: c:\d79.log
Machine: n109
Parameter Set: 2.2
Index: 123400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER792820421
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d80
Description:
Directory: c:\d80.log
Machine: n109
Parameter Set: 2.2
Index: 123700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER802842390
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d81
Description:
Directory: c:\d81.log
Machine: n110
Parameter Set: 2.2
Index: 124000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER812851328
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d82
Description:
Directory: c:\d82.log
Machine: n110
Parameter Set: 2.2
Index: 124300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER823364343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d83
Description:
Directory: c:\d83.log
Machine: n110
Parameter Set: 2.2
Index: 124600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER833381656
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d84
Description:
Directory: c:\d84.log
Machine: n110
Parameter Set: 2.2
Index: 124900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER843392562
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d85
Description:
Directory: c:\d85.log
Machine: n111
Parameter Set: 2.2
Index: 125200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER8554757562
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d86
Description:
Directory: c:\d86.log
Machine: n111
Parameter Set: 2.2
Index: 125500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER8654864968
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d87
Description:
Directory: c:\d87.log
Machine: n111
Parameter Set: 2.2
Index: 125800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER8754901734
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d88
Description:
Directory: c:\d88.log
Machine: n111
Parameter Set: 2.2
Index: 126100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER8855059343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d89
```

```
Description:
Directory: c:\d89.log
Machine: n112
Parameter Set: 2.2
Index: 126400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER8955092343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d90
Description:
Directory: c:\d90.log
Machine: n112
Parameter Set: 2.2
Index: 126700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9055486578
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d91
Description:
Directory: c:\d91.log
Machine: n112
Parameter Set: 2.2
Index: 127000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9155534031
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d92
Description:
Directory: c:\d92.log
Machine: n112
Parameter Set: 2.2
Index: 127300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9255579359
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d93
Description:
Directory: c:\d93.log
Machine: n113
Parameter Set: 2.2
Index: 127600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9355620406
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d94
Description:
Directory: c:\d94.log
Machine: n113
Parameter Set: 2.2
Index: 127900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9455653265
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d95
Description:
Directory: c:\d95.log
Machine: n113
Parameter Set: 2.2
Index: 128200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9555683343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d96
Description:
Directory: c:\d96.log
Machine: n113
Parameter Set: 2.2
Index: 128500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9655715281
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d97
Description:
Directory: c:\d97.log
Machine: n114
Parameter Set: 2.2
Index: 128800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9791039015
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d98
Description:
Directory: c:\d98.log
Machine: n114
Parameter Set: 2.2
Index: 129100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9891080187
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d99
Description:
Directory: c:\d99.log
Machine: n114
Parameter Set: 2.2
Index: 129400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER9991094312
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d100
Description:
Directory: c:\d100.log
```

```
Machine: n114
Parameter Set: 2.2
Index: 129700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10091114343
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d101
Description:
Directory: c:\d101.log
Machine: n115
Parameter Set: 2.2
Index: 130000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10191182218
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d102
Description:
Directory: c:\d102.log
Machine: n115
Parameter Set: 2.2
Index: 130300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10291195265
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d103
Description:
Directory: c:\d103.log
Machine: n115
Parameter Set: 2.2
Index: 130600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10391210015
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d104
Description:
Directory: c:\d104.log
Machine: n115
Parameter Set: 2.2
Index: 130900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10491224609
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d105
Description:
Directory: c:\d105.log
Machine: n116
Parameter Set: 2.2
Index: 131200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10591253062
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d106
Description:
Directory: c:\d106.log
Machine: n116
Parameter Set: 2.2
Index: 131500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10691266500
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d107
Description:
Directory: c:\d107.log
Machine: n116
Parameter Set: 2.2
Index: 131800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10791279375
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d108
Description:
Directory: c:\d108.log
Machine: n116
Parameter Set: 2.2
Index: 132100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10891300515
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d109
Description:
Directory: c:\d109.log
Machine: n117
Parameter Set: 2.2
Index: 132400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER10991349968
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d110
Description:
Directory: c:\d110.log
Machine: n117
Parameter Set: 2.2
Index: 132700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11091369218
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d111
Description:
Directory: c:\d111.log
Machine: n117
Parameter Set: 2.2
```

```
Index: 133000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11191391000
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d112
Description:
Directory: c:\d112.log
Machine: n117
Parameter Set: 2.2
Index: 133300000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11291408671
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d113
Description:
Directory: c:\d113.log
Machine: n118
Parameter Set: 2.2
Index: 133600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11391437843
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d114
Description:
Directory: c:\d114.log
Machine: n118
Parameter Set: 2.2
Index: 133900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11491503046
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d115
Description:
Directory: c:\d115.log
Machine: n118
Parameter Set: 2.2
Index: 134200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11591544156
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d116
Description:
Directory: c:\d116.log
Machine: n118
Parameter Set: 2.2
Index: 134500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11691602328
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d117
Description:
Directory: c:\d117.log
Machine: n119
Parameter Set: 2.2
Index: 134800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11791774937
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d118
Description:
Directory: c:\d118.log
Machine: n119
Parameter Set: 2.2
Index: 135100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11891802062
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d119
Description:
Directory: c:\d119.log
Machine: n119
Parameter Set: 2.2
Index: 135400000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER11991877296
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d120
Description:
Directory: c:\d120.log
Machine: n119
Parameter Set: 2.2
Index: 135700000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER12091903921
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d121
Description:
Directory: c:\d121.log
Machine: n120
Parameter Set: 2.2
Index: 136000000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER12191918468
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d122
Description:
Directory: c:\d122.log
Machine: n120
Parameter Set: 2.2
Index: 136300000
Seed: 4678
```

```
Configured Users: 6560
Pipe Name: DRIVER12291955406
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d123
Description:
Directory: c:\d123.log
Machine: n120
Parameter Set: 2.2
Index: 136600000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER12391970531
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d124
Description:
Directory: c:\d124.log
Machine: n120
Parameter Set: 2.2
Index: 136900000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER12491983140
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

Name: d125
Description:
Directory: c:\d125.log
Machine: n121
Parameter Set: 2.2
Index: 137200000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER12591999843
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 0
Additional Options:

Name: d126
Description:
Directory: c:\d126.log
Machine: n121
Parameter Set: 2.2
Index: 137500000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER12692036203
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 1
Additional Options:

Name: d127
Description:
Directory: c:\d127.log
Machine: n121
Parameter Set: 2.2
Index: 137800000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER12792054906
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 4
Additional Options:

Name: d128
Description:
Directory: c:\d128.log
Machine: n121
Parameter Set: 2.2
Index: 138100000
Seed: 4678
Configured Users: 6560
Pipe Name: DRIVER12892076843
Connect Rate: 10
Start Rate: 0
Max. Concurrency: 0
Concurrency Rate: 0
CLIENT_NURAND: 25
CPU: 5
Additional Options:

        Number of User groups: 128

Driver Engine: d1
IIS Server: vcr1
SQL Server:
tcp:130.168.209.71,2001
Database: tpcc
User: sa
Protocol: HTML
w_id Range: 1 - 656
w_id Min Warehouse: 1
w_id Max Warehouse: 83968
Scale: Normal
User Count: 6560
District id: 1
Scale Down: No

Driver Engine: d2
IIS Server: vcr1
SQL Server:
tcp:130.168.209.71,2001
Database: tpcc
User: sa
Protocol: HTML
w_id Range: 657 - 1312
w_id Min Warehouse: 1
w_id Max Warehouse: 83968
Scale: Normal
User Count: 6560
District id: 1
Scale Down: No

Driver Engine: d3
IIS Server: vcr1
SQL Server:
tcp:130.168.209.71,2001
Database: tpcc
User: sa
Protocol: HTML
w_id Range: 1313 - 1968
w_id Min Warehouse: 1
w_id Max Warehouse: 83968
Scale: Normal
User Count: 6560
District id: 1
Scale Down: No

Driver Engine: d4
IIS Server: vcr1
SQL Server:
tcp:130.168.209.71,2001
Database: tpcc
User: sa
Protocol: HTML
w_id Range: 1969 - 2624
w_id Min Warehouse: 1
w_id Max Warehouse: 83968
Scale: Normal
User Count: 6560
District id: 1
Scale Down: No

Driver Engine: d5
IIS Server: vcr2
SQL Server:
tcp:130.168.209.71,2001
Database: tpcc
User: sa
Protocol: HTML
w_id Range: 2625 - 3280
w_id Min Warehouse: 1
w_id Max Warehouse: 83968
Scale: Normal
User Count: 6560
District id: 1
Scale Down: No
```

```
                    Driver Engine: d6
                    IIS Server: vcr2
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 3281 - 3936
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d7
                    IIS Server: vcr2
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 3937 - 4592
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d8
                    IIS Server: vcr2
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 4593 - 5248
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d9
                    IIS Server: vcr33
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 5249 - 5904
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d10
                    IIS Server: vcr33
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 5905 - 6560
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d11
                    IIS Server: vcr33
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 6561 - 7216
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d12
                    IIS Server: vcr33
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 7217 - 7872
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d13
                    IIS Server: vcr4
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 7873 - 8528
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d14
                    IIS Server: vcr4
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 8529 - 9184
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d15
                    IIS Server: vcr4
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 9185 - 9840
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d16
                    IIS Server: vcr4
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 9841 - 10496
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d17
                    IIS Server: vcr5
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 10497 - 11152
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d18
                    IIS Server: vcr5
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 11153 - 11808
```

```
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d19
                    IIS Server: vcr5
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 11809 - 12464
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d20
                    IIS Server: vcr5
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 12465 - 13120
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d21
                    IIS Server: vcr6
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 13121 - 13776
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d22
                    IIS Server: vcr6
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 13777 - 14432
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d23
                    IIS Server: vcr6
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 14433 - 15088
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d24
                    IIS Server: vcr6
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 15089 - 15744
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d25
                    IIS Server: vcr7
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 15745 - 16400
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d26
                    IIS Server: vcr7
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 16401 - 17056
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d27
                    IIS Server: vcr7
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 17057 - 17712
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d28
                    IIS Server: vcr7
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 17713 - 18368
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d29
                    IIS Server: vcr8
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 18369 - 19024
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d30
                    IIS Server: vcr8
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 19025 - 19680
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d31
                    IIS Server: vcr8
```

```
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 19681 - 20336
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d32
                    IIS Server: vcr8
                    SQL Server:
tcp:130.168.209.71,2001
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 20337 - 20992
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d33
                    IIS Server: vcr9
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 20993 - 21648
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d34
                    IIS Server: vcr9
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 21649 - 22304
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d35
                    IIS Server: vcr9
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 22305 - 22960
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d36
                    IIS Server: vcr9
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 22961 - 23616
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d37
                    IIS Server: vcr34
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 23617 - 24272
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d38
                    IIS Server: vcr34
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 24273 - 24928
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d39
                    IIS Server: vcr34
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 24929 - 25584
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d40
                    IIS Server: vcr34
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 25585 - 26240
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d41
                    IIS Server: vcr11
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 26241 - 26896
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d42
                    IIS Server: vcr11
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 26897 - 27552
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d43
                    IIS Server: vcr11
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 27553 - 28208
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
```

```
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d44
                    IIS Server: vcr11
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 28209 - 28864
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d45
                    IIS Server: vcr12
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 28865 - 29520
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d46
                    IIS Server: vcr12
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 29521 - 30176
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d47
                    IIS Server: vcr12
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 30177 - 30832
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d48
                    IIS Server: vcr12
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 30833 - 31488
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d49
                    IIS Server: vcr13
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 31489 - 32144
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d50
                    IIS Server: vcr13
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 32145 - 32800
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d51
                    IIS Server: vcr13
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 32801 - 33456
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d52
                    IIS Server: vcr13
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 33457 - 34112
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d53
                    IIS Server: vcr14
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 34113 - 34768
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d54
                    IIS Server: vcr14
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 34769 - 35424
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d55
                    IIS Server: vcr14
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 35425 - 36080
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d56
                    IIS Server: vcr14
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
```

```
                    User: sa
                    Protocol: HTML
                    w_id Range: 36081 - 36736
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d57
                    IIS Server: vcr15
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 36737 - 37392
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d58
                    IIS Server: vcr15
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 37393 - 38048
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d59
                    IIS Server: vcr15
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 38049 - 38704
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d60
                    IIS Server: vcr15
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 38705 - 39360
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d61
                    IIS Server: vcr16
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 39361 - 40016
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d62
                    IIS Server: vcr16
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 40017 - 40672
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d63
                    IIS Server: vcr16
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 40673 - 41328
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d64
                    IIS Server: vcr16
                    SQL Server:
tcp:130.168.209.72,2002
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 41329 - 41984
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d65
                    IIS Server: vcr17
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 41985 - 42640
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d66
                    IIS Server: vcr17
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 42641 - 43296
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d67
                    IIS Server: vcr17
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 43297 - 43952
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d68
                    IIS Server: vcr17
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 43953 - 44608
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No
```

```
                    Driver Engine: d69
                    IIS Server: vcr18
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 44609 - 45264
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d70
                    IIS Server: vcr18
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 45265 - 45920
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d71
                    IIS Server: vcr18
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 45921 - 46576
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d72
                    IIS Server: vcr18
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 46577 - 47232
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d73
                    IIS Server: vcr19
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 47233 - 47888
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d74
                    IIS Server: vcr19
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 47889 - 48544
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d75
                    IIS Server: vcr19
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 48545 - 49200
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d76
                    IIS Server: vcr19
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 49201 - 49856
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d77
                    IIS Server: vcr20
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 49857 - 50512
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d78
                    IIS Server: vcr20
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 50513 - 51168
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d79
                    IIS Server: vcr20
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 51169 - 51824
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d80
                    IIS Server: vcr20
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 51825 - 52480
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d81
                    IIS Server: vcr21
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 52481 - 53136
```

```
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d82
                    IIS Server: vcr21
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 53137 - 53792
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d83
                    IIS Server: vcr21
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 53793 - 54448
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d84
                    IIS Server: vcr21
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 54449 - 55104
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d85
                    IIS Server: vcr22
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 55105 - 55760
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d86
                    IIS Server: vcr22
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 55761 - 56416
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d87
                    IIS Server: vcr22
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 56417 - 57072
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d88
                    IIS Server: vcr22
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 57073 - 57728
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d89
                    IIS Server: vcr23
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 57729 - 58384
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d90
                    IIS Server: vcr23
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 58385 - 59040
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d91
                    IIS Server: vcr23
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 59041 - 59696
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d92
                    IIS Server: vcr23
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 59697 - 60352
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d93
                    IIS Server: vcr24
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 60353 - 61008
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d94
                    IIS Server: vcr24
```

```
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 61009 - 61664
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d95
                    IIS Server: vcr24
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 61665 - 62320
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d96
                    IIS Server: vcr24
                    SQL Server:
tcp:130.168.209.73,2003
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 62321 - 62976
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d97
                    IIS Server: vcr25
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 62977 - 63632
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d98
                    IIS Server: vcr25
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 63633 - 64288
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d99
                    IIS Server: vcr25
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 64289 - 64944
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d100
                    IIS Server: vcr25
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 64945 - 65600
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d101
                    IIS Server: vcr34
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 65601 - 66256
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d102
                    IIS Server: vcr34
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 66257 - 66912
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d103
                    IIS Server: vcr34
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 66913 - 67568
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d104
                    IIS Server: vcr34
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 67569 - 68224
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d105
                    IIS Server: vcr27
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 68225 - 68880
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d106
                    IIS Server: vcr27
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 68881 - 69536
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
```

```
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d107
                    IIS Server: vcr27
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 69537 - 70192
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d108
                    IIS Server: vcr27
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 70193 - 70848
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d109
                    IIS Server: vcr28
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 70849 - 71504
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d110
                    IIS Server: vcr28
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 71505 - 72160
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d111
                    IIS Server: vcr28
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 72161 - 72816
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d112
                    IIS Server: vcr28
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 72817 - 73472
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d113
                    IIS Server: vcr29
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 73473 - 74128
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d114
                    IIS Server: vcr29
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 74129 - 74784
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d115
                    IIS Server: vcr29
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 74785 - 75440
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d116
                    IIS Server: vcr29
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 75441 - 76096
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d117
                    IIS Server: vcr30
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 76097 - 76752
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d118
                    IIS Server: vcr30
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 76753 - 77408
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d119
                    IIS Server: vcr30
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
```

```
                    User: sa
                    Protocol: HTML
                    w_id Range: 77409 - 78064
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d120
                    IIS Server: vcr30
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 78065 - 78720
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d121
                    IIS Server: vcr31
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 78721 - 79376
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d122
                    IIS Server: vcr31
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 79377 - 80032
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d123
                    IIS Server: vcr31
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 80033 - 80688
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d124
                    IIS Server: vcr31
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 80689 - 81344
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d125
                    IIS Server: vcr32
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 81345 - 82000
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d126
                    IIS Server: vcr32
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 82001 - 82656
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d127
                    IIS Server: vcr32
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 82657 - 83312
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

                    Driver Engine: d128
                    IIS Server: vcr32
                    SQL Server:
tcp:130.168.209.74,2004
                    Database: tpcc
                    User: sa
                    Protocol: HTML
                    w_id Range: 83313 - 83968
                    w_id Min Warehouse: 1
                    w_id Max Warehouse: 83968
                    Scale: Normal
                    User Count: 6560
                    District id: 1
                    Scale Down: No

            Number of Parameter Sets: 61


                    ~Default
                    Default Parameter Set
                                                Txn     Think
Key     RT      RT      Menu
                                               Weight  Time
Time    Delay   Fence   Delay
                    New Order          10.00
12.05       18.01       0.10       5.00       0.10
                    Payment            10.00
12.05        3.01       0.10       5.00       0.10
                    Delivery            1.00
5.05         2.01       0.10       5.00       0.10
                    Stock Level         1.00
5.05         2.01       0.10      20.00       0.10
                    Order Status        1.00
10.05        2.01       0.10       5.00       0.10

                    Tuned Distribution

                                                Txn     Think
Key     RT      RT      Menu
                                               Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
12.05       18.01       0.10       5.00       0.10
                    Payment            43.10
12.05        3.01       0.10       5.00       0.10
                    Delivery            4.05
5.05         2.01       0.10       5.00       0.10
                    Stock Level         4.05
5.05         2.01       0.10      20.00       0.10
                    Order Status        4.05
10.05        2.01       0.10       5.00       0.10

                    No Think

                                                Txn     Think
Key     RT      RT      Menu
                                               Weight  Time
Time    Delay   Fence   Delay
```

```
                    New Order          10.00
0.00          0.00      0.00       5.00       0.00
                    Payment            10.00
0.00          0.00      0.00       5.00       0.00
                    Delivery            1.00
0.00          0.00      0.00       5.00       0.00
                    Stock Level         1.00
0.00          0.00      0.00      20.00       0.00
                    Order Status        1.00
0.00          0.00      0.00       5.00       0.00

                    0.95

                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
13.00       18.01      0.10       5.00       0.10
                    Payment            43.10
13.00        3.01      0.10       5.00       0.10
                    Delivery            4.05
6.00         2.01      0.10       5.00       0.10
                    Stock Level         4.05
6.00         2.01      0.10      20.00       0.10
                    Order Status        4.05
11.00        2.01      0.10       5.00       0.10

                    0.9

                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.83
16.00       18.01      0.10       5.00       0.10
                    Payment            43.05
16.00        3.01      0.10       5.00       0.10
                    Delivery            4.04
9.00         2.01      0.10       5.00       0.10
                    Stock Level         4.04
9.00         2.01      0.10      20.00       0.10
                    Order Status        4.04
14.00        2.01      0.10       5.00       0.10

                    3

                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
36.15        0.00      0.10       5.00       0.10
                    Payment            43.10
36.15        0.00      0.10       5.00       0.10
                    Delivery            4.05
15.15        0.00      0.10       5.00       0.10
                    Stock Level         4.05
15.15        0.00      0.10      20.00       0.10
                    Order Status        4.05
30.15        0.00      0.10       5.00       0.10

                    4

                    4.0 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
48.20       18.01      0.10       5.00       0.10
                    Payment            43.10
48.20        3.01      0.10       5.00       0.10
                    Delivery            4.05
20.20        2.01      0.10       5.00       0.10
                    Stock Level         4.05
20.20        2.01      0.10      20.00       0.10
                    Order Status        4.05
40.20        2.01      0.10       5.00       0.10

                    3.8
                    3.8 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
45.70       18.01      0.10       5.00       0.10
                    Payment            43.10
45.70        3.01      0.10       5.00       0.10
                    Delivery            4.05
19.10        2.01      0.10       5.00       0.10
                    Stock Level         4.05
19.10        2.01      0.10      20.00       0.10
                    Order Status        4.05
38.10        2.01      0.10       5.00       0.10

                    3.6
                    3.6 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
43.30       18.01      0.10       5.00       0.10
                    Payment            43.10
43.30        3.01      0.10       5.00       0.10
                    Delivery            4.05
18.10        2.01      0.10       5.00       0.10
                    Stock Level         4.05
18.10        2.01      0.10      20.00       0.10
                    Order Status        4.05
36.18        2.01      0.10       5.00       0.10

                    3.4
                    3.4 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
40.90       18.01      0.10       5.00       0.10
                    Payment            43.10
40.90        3.01      0.10       5.00       0.10
                    Delivery            4.05
17.10        2.01      0.10       5.00       0.10
                    Stock Level         4.05
17.10        2.01      0.10      20.00       0.10
                    Order Status        4.05
17.10        2.01      0.10       5.00       0.10

                    3.2
                    3.2 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
38.50       18.01      0.10       5.00       0.10
                    Payment            43.10
38.50        3.01      0.10       5.00       0.10
                    Delivery            4.05
16.10        2.01      0.10       5.00       0.10
                    Stock Level         4.05
16.10        2.01      0.10      20.00       0.10
                    Order Status        4.05
32.10        2.01      0.10       5.00       0.10

                    2.8
                    2.8 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
33.74       18.01      0.10       5.00       0.10
                    Payment            43.10
33.74        3.01      0.10       5.00       0.10
                    Delivery            4.05
14.14        2.01      0.10       5.00       0.10
                    Stock Level         4.05
14.14        2.01      0.10      20.00       0.10
                    Order Status        4.05
28.14        2.01      0.10       5.00       0.10

                    2.6
                    2.6 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
31.30       18.01      0.10       5.00       0.10
                    Payment            43.10
31.30        3.01      0.10       5.00       0.10
                    Delivery            4.05
13.10        2.01      0.10       5.00       0.10
                    Stock Level         4.05
13.10        2.01      0.10      20.00       0.10
                    Order Status        4.05
26.10        2.01      0.10       5.00       0.10

                    2.4
                    2.4 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
```

```
                    New Order          44.75
28.90       18.01       0.10       5.00       0.10
                    Payment           43.10
28.90        3.01       0.10       5.00       0.10
                    Delivery           4.05
12.10        2.01       0.10       5.00       0.10
                    Stock Level        4.05
12.10        2.01       0.10      20.00       0.10
                    Order Status       4.05
24.10        2.01       0.10       5.00       0.10

                    2.2
                    2.2 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
28.90       18.01       0.10       5.00       0.10
                    Payment           43.10
28.90        3.01       0.10       5.00       0.10
                    Delivery           4.05
12.10        2.01       0.10       5.00       0.10
                    Stock Level        4.05
12.10        2.01       0.10      20.00       0.10
                    Order Status       4.05
24.12        2.01       0.10       5.00       0.10

                    2
                    2.0 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
24.10       18.01       0.10       5.00       0.10
                    Payment           43.10
24.10        3.01       0.10       5.00       0.10
                    Delivery           4.05
10.10        2.01       0.10       5.00       0.10
                    Stock Level        4.05
10.10        2.01       0.10      20.00       0.10
                    Order Status       4.05
20.10        2.01       0.10       5.00       0.10

                    5
                    5.0 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
60.25       18.01       0.10       5.00       0.10
                    Payment           43.10
60.25        3.01       0.10       5.00       0.10
                    Delivery           4.05
25.25        2.01       0.10       5.00       0.10
                    Stock Level        4.05
25.25        2.01       0.10      20.00       0.10
                    Order Status       4.05
50.25        2.01       0.10       5.00       0.10

                    4.5
                    4.5 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
54.20       18.01       0.10       5.00       0.10
                    Payment           43.10
54.20        3.01       0.10       5.00       0.10
                    Delivery           4.05
22.70        2.01       0.10       5.00       0.10
                    Stock Level        4.05
22.70        2.01       0.10      20.00       0.10
                    Order Status       4.05
45.20        2.01       0.10       5.00       0.10

                    3.5
                    3.5 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
42.10       18.01       0.10       5.00       0.10
                    Payment           43.10
42.10        3.01       0.10       5.00       0.10
                    Delivery           4.05
17.60        2.01       0.10       5.00       0.10
                    Stock Level        4.05
17.60        2.01       0.10      20.00       0.10
                    Order Status       4.05
35.10        2.01       0.10       5.00       0.10

                    1.8
                    1.8 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
21.60       18.01       0.10       5.00       0.10
                    Payment           43.10
21.60        3.01       0.10       5.00       0.10
                    Delivery           4.05
9.09         2.01       0.10       5.00       0.10
                    Stock Level        4.05
9.09         2.01       0.10      20.00       0.10
                    Order Status       4.05
18.09        2.01       0.10       5.00       0.10

                    4.2
                    4.2 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
54.20       18.01       0.10       5.00       0.10
                    Payment           43.10
54.20        3.01       0.10       5.00       0.10
                    Delivery           4.05
22.70        2.01       0.10       5.00       0.10
                    Stock Level        4.05
22.70        2.01       0.10      20.00       0.10
                    Order Status       4.05
45.20        2.01       0.10       5.00       0.10

                    1.6
                    1.6 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.75
19.20       18.01       0.10       5.00       0.10
                    Payment           43.10
19.20        3.01       0.10       5.00       0.10
                    Delivery           4.05
8.08         2.01       0.10       5.00       0.10
                    Stock Level        4.05
8.08         2.01       0.10      20.00       0.10
                    Order Status       4.05
16.08        2.01       0.10       5.00       0.10

                    1.4
                    1.4 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.91
16.87       18.01       0.10       5.00       0.10
                    Payment           43.02
16.87        3.01       0.10       5.00       0.10
                    Delivery           4.02
7.07         2.01       0.10       5.00       0.10
                    Stock Level        4.02
7.07         2.01       0.10      20.00       0.10
                    Order Status       4.03
14.07        2.01       0.10       5.00       0.10

                    1.2
                    1.2 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.94
14.46       18.01       0.10       5.00       0.10
                    Payment           43.02
14.46        3.01       0.10       5.00       0.10
                    Delivery           4.01
6.06         2.01       0.10       5.00       0.10
                    Stock Level        4.01
6.06         2.01       0.10      20.00       0.10
                    Order Status       4.02
12.06        2.01       0.10       5.00       0.10

                    3.5
                    3.5 tt
                                        Txn     Think
Key     RT      RT      Menu
                                     Weight  Time
Time    Delay   Fence   Delay
```

```
                    New Order          44.75
42.10       18.01       0.10       5.00       0.10
                    Payment            43.10
42.10        3.01       0.10       5.00       0.10
                    Delivery            4.05
17.60        2.01       0.10       5.00       0.10
                    Stock Level         4.05
17.60        2.01       0.10      20.00       0.10
                    Order Status        4.05
35.10        2.01       0.10       5.00       0.10


                    1.9
                    1.9 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
22.89       18.01       0.10       5.00       0.10
                    Payment            43.10
22.89        3.01       0.10       5.00       0.10
                    Delivery            4.05
9.59         2.01       0.10       5.00       0.10
                    Stock Level         4.05
9.59         2.01       0.10      20.00       0.10
                    Order Status        4.05
19.09        2.01       0.10       5.00       0.10


                    1.1
                    1.1 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.91
13.25       18.01       0.10       5.00       0.10
                    Payment            43.02
13.25        3.01       0.10       5.00       0.10
                    Delivery            4.02
5.55         2.01       0.10       5.00       0.10
                    Stock Level         4.02
5.55         2.01       0.10      20.00       0.10
                    Order Status        4.03
11.05        2.01       0.10       5.00       0.10


                    1.05 better
                    1.05 tt better
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.91
12.65       18.01       0.10       5.00       0.10
                    Payment            43.02
12.65        3.01       0.10       5.00       0.10
                    Delivery            4.02
5.30         2.01       0.10       5.00       0.10
                    Stock Level         4.02
5.30         2.01       0.10      20.00       0.10
                    Order Status        4.03
10.55        2.01       0.10       5.00       0.10


                    1.09
                    1.09 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.83
13.13       18.01       0.10       5.00       0.10
                    Payment            43.05
13.13        3.01       0.10       5.00       0.10
                    Delivery            4.04
5.50         2.01       0.10       5.00       0.10
                    Stock Level         4.04
5.50         2.01       0.10      20.00       0.10
                    Order Status        4.04
10.95        2.01       0.10       5.00       0.10


                    1.08
                    1.08 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.83
13.01       18.01       0.10       5.00       0.10
                    Payment            43.05
13.01        3.01       0.10       5.00       0.10
                    Delivery            4.04
5.45         2.01       0.10       5.00       0.10
                    Stock Level         4.04
5.45         2.01       0.10      20.00       0.10
                    Order Status        4.04
10.85        2.01       0.10       5.00       0.10


                    1.07
                    1.07 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.83
12.89       18.01       0.10       5.00       0.10
                    Payment            43.05
12.89        3.01       0.10       5.00       0.10
                    Delivery            4.04
5.40         2.01       0.10       5.00       0.10
                    Stock Level         4.04
5.40         2.01       0.10      20.00       0.10
                    Order Status        4.04
10.75        2.01       0.10       5.00       0.10


                    1.06
                    1.06 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.83
12.77       18.01       0.10       5.00       0.10
                    Payment            43.05
12.77        3.01       0.10       5.00       0.10
                    Delivery            4.04
5.35         2.01       0.10       5.00       0.10
                    Stock Level         4.04
5.35         2.01       0.10      20.00       0.10
                    Order Status        4.04
10.65        2.01       0.10       5.00       0.10


                    1.15
                    1.15 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.75
13.85       18.01       0.10       5.00       0.10
                    Payment            43.10
13.85        3.01       0.10       5.00       0.10
                    Delivery            4.05
5.80         2.01       0.10       5.00       0.10
                    Stock Level         4.05
5.80         2.01       0.10      20.00       0.10
                    Order Status        4.05
11.55        2.01       0.10       5.00       0.10


                    1.25
                    1.25 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.83
15.06       18.01       0.10       5.00       0.10
                    Payment            43.05
15.06        3.01       0.10       5.00       0.10
                    Delivery            4.04
6.31         2.01       0.10       5.00       0.10
                    Stock Level         4.04
6.31         2.01       0.10      20.00       0.10
                    Order Status        4.04
12.56        2.01       0.10       5.00       0.10


                    1.3
                    1.3 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
                    New Order          44.83
15.66       18.01       0.10       5.00       0.10
                    Payment            43.05
15.66        3.01       0.10       5.00       0.10
                    Delivery            4.04
6.56         2.01       0.10       5.00       0.10
                    Stock Level         4.04
6.56         2.01       0.10      20.00       0.10
                    Order Status        4.04
13.06        2.01       0.10       5.00       0.10


                    1.12
                    1.12 tt
                                          Txn      Think
Key      RT       RT       Menu
                                       Weight   Time
Time     Delay    Fence    Delay
```

```
                    New Order          44.75
13.49       18.01       0.10       5.00       0.10
                    Payment            43.10
13.49        3.01       0.10       5.00       0.10
                    Delivery            4.05
5.65         2.01       0.10       5.00       0.10
                    Stock Level         4.05
5.65         2.01       0.10      20.00       0.10
                    Order Status        4.05
11.25        2.01       0.10       5.00       0.10

                    1.18
                    1.18 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.75
14.21       18.01       0.10       5.00       0.10
                    Payment            43.10
14.21        3.01       0.10       5.00       0.10
                    Delivery            4.05
5.95         2.01       0.10       5.00       0.10
                    Stock Level         4.05
5.95         2.01       0.10      20.00       0.10
                    Order Status        4.05
11.85        2.01       0.10       5.00       0.10

                    1.22
                    1.22 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.75
14.70       18.01       0.10       5.00       0.10
                    Payment            43.10
14.70        3.01       0.10       5.00       0.10
                    Delivery            4.05
6.16         2.01       0.10       5.00       0.10
                    Stock Level         4.05
6.16         2.01       0.10      20.00       0.10
                    Order Status        4.05
12.26        2.01       0.10       5.00       0.10

                    1.28
                    1.28 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.75
15.42       18.01       0.10       5.00       0.10
                    Payment            43.10
15.42        3.01       0.10       5.00       0.10
                    Delivery            4.05
6.46         2.01       0.10       5.00       0.10
                    Stock Level         4.05
6.46         2.01       0.10      20.00       0.10
                    Order Status        4.05
12.86        2.01       0.10       5.00       0.10

                    1.04
                    1.04 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.83
12.53       18.01       0.10       5.00       0.10
                    Payment            43.05
12.53        3.01       0.10       5.00       0.10
                    Delivery            4.04
5.25         2.01       0.10       5.00       0.10
                    Stock Level         4.04
5.25         2.01       0.10      20.00       0.10
                    Order Status        4.04
10.45        2.01       0.10       5.00       0.10

                    1.03
                    1.03 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.83
12.41       18.01       0.10       5.00       0.10
                    Payment            43.05
12.41        3.01       0.10       5.00       0.10
                    Delivery            4.04
5.20         2.01       0.10       5.00       0.10
                    Stock Level         4.04
5.20         2.01       0.10      20.00       0.10
                    Order Status        4.04
10.35        2.01       0.10       5.00       0.10

                    1.02
                    1.02 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.83
12.29       18.01       0.10       5.00       0.10
                    Payment            43.05
12.29        3.01       0.10       5.00       0.10
                    Delivery            4.04
5.15         2.01       0.10       5.00       0.10
                    Stock Level         4.04
5.15         2.01       0.10      20.00       0.10
                    Order Status        4.04
10.25        2.01       0.10       5.00       0.10

                    1.01
                    1.01 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.83
12.17       18.01       0.10       5.00       0.10
                    Payment            43.05
12.17        3.01       0.10       5.00       0.10
                    Delivery            4.04
5.10         2.01       0.10       5.00       0.10
                    Stock Level         4.04
5.10         2.01       0.10      20.00       0.10
                    Order Status        4.04
10.15        2.01       0.10       5.00       0.10

                    5.5
                    5.5 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.83
66.28       18.01       0.10       5.00       0.10
                    Payment            43.05
66.28        3.01       0.10       5.00       0.10
                    Delivery            4.04
27.77        2.01       0.10       5.00       0.10
                    Stock Level         4.04
27.77        2.01       0.10      20.00       0.10
                    Order Status        4.04
55.27        2.01       0.10       5.00       0.10

                    6
                    6.0 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.83
72.30       18.01       0.10       5.00       0.10
                    Payment            43.05
72.30        3.01       0.10       5.00       0.10
                    Delivery            4.04
30.30        2.01       0.10       5.00       0.10
                    Stock Level         4.04
30.30        2.01       0.10      20.00       0.10
                    Order Status        4.04
60.30        2.01       0.10       5.00       0.10

                    6.5
                    6.5 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
                    New Order          44.83
79.53       18.01       0.10       5.00       0.10
                    Payment            43.05
79.53        3.01       0.10       5.00       0.10
                    Delivery            4.04
33.33        2.01       0.10       5.00       0.10
                    Stock Level         4.04
33.33        2.01       0.10      20.00       0.10
                    Order Status        4.04
66.33        2.01       0.10       5.00       0.10

                    7
                    7.0 tt
                                          Txn      Think
Key      RT      RT      Menu
                                       Weight   Time
Time     Delay   Fence   Delay
```

```
                    New Order          44.83
84.35       18.01       0.10       5.00       0.10
                    Payment            43.05
84.35        3.01       0.10       5.00       0.10
                    Delivery            4.04
35.35        2.01       0.10       5.00       0.10
                    Stock Level         4.04
35.35        2.01       0.10      20.00       0.10
                    Order Status        4.04
70.35        2.01       0.10       5.00       0.10

                    7.5
                    7.5 tt
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.83
90.38       18.01       0.10       5.00       0.10
                    Payment            43.05
90.38        3.01       0.10       5.00       0.10
                    Delivery            4.04
37.88        2.01       0.10       5.00       0.10
                    Stock Level         4.04
37.88        2.01       0.10      20.00       0.10
                    Order Status        4.04
75.38        2.01       0.10       5.00       0.10

                    8
                    8.0 tt
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.83
96.40       18.01       0.10       5.00       0.10
                    Payment            43.05
96.40        3.01       0.10       5.00       0.10
                    Delivery            4.04
40.40        2.01       0.10       5.00       0.10
                    Stock Level         4.04
40.40        2.01       0.10      20.00       0.10
                    Order Status        4.04
80.40        2.01       0.10       5.00       0.10

                    8.5
                    8.5 tt
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.83
102.43      18.01       0.10       5.00       0.10
                    Payment            43.05
192.43       3.01       0.10       5.00       0.10
                    Delivery            4.04
42.92        2.01       0.10       5.00       0.10
                    Stock Level         4.04
42.92        2.01       0.10      20.00       0.10
                    Order Status        4.04
85.42        2.01       0.10       5.00       0.10

                    9
                    9.0 tt
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.83
108.45      18.01       0.10       5.00       0.10
                    Payment            43.05
108.45       3.01       0.10       5.00       0.10
                    Delivery            4.04
45.45        2.01       0.10       5.00       0.10
                    Stock Level         4.04
45.45        2.01       0.10      20.00       0.10
                    Order Status        4.04
90.45        2.01       0.10       5.00       0.10

                    9.5
                    9.5 tt
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.83
114.47      18.01       0.10       5.00       0.10
                    Payment            43.05
114.47       3.01       0.10       5.00       0.10
                    Delivery            4.04
47.98        2.01       0.10       5.00       0.10
                    Stock Level         4.04
47.98        2.01       0.10      20.00       0.10
                    Order Status        4.04
95.47        2.01       0.10       5.00       0.10

                    10
                    10 tt
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.83
120.50      18.01       0.10       5.00       0.10
                    Payment            43.05
120.50       3.01       0.10       5.00       0.10
                    Delivery            4.04
50.50        2.01       0.10       5.00       0.10
                    Stock Level         4.04
50.50        2.01       0.10      20.00       0.10
                    Order Status        4.04
100.50       2.01       0.10       5.00       0.10

                    1.02 better
                    1.02 more agressive
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.91
12.05       18.01       0.10       5.00       0.10
                    Payment            43.02
12.05        3.01       0.10       5.00       0.10
                    Delivery            4.02
5.05         2.01       0.10       5.00       0.10
                    Stock Level         4.02
5.05         2.01       0.10      20.00       0.10
                    Order Status        4.03
10.05        2.01       0.10       5.00       0.10

                    1.01 better
                    1.01 more aggressive
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.91
12.17       18.01       0.10       5.00       0.10
                    Payment            43.02
12.17        3.01       0.10       5.00       0.10
                    Delivery            4.02
5.10         2.01       0.10       5.00       0.10
                    Stock Level         4.02
5.10         2.01       0.10      20.00       0.10
                    Order Status        4.03
10.15        2.01       0.10       5.00       0.10

                    1.001
                    1.001
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.94
12.06       18.01       0.10       5.00       0.10
                    Payment            43.03
12.06        3.01       0.10       5.00       0.10
                    Delivery            4.01
5.06         2.01       0.10       5.00       0.10
                    Stock Level         4.01
5.06         2.01       0.10      20.00       0.10
                    Order Status        4.01
10.06        2.01       0.10       5.00       0.10

                    FullSpeed
                    1.000 tt
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
                    New Order          44.91
12.05       18.01       0.10       5.00       0.10
                    Payment            43.02
12.05        3.01       0.10       5.00       0.10
                    Delivery            4.02
5.05         2.01       0.10       5.00       0.10
                    Stock Level         4.02
5.05         2.01       0.10      20.00       0.10
                    Order Status        4.03
10.05        2.01       0.10       5.00       0.10

                    1.003 best
                    1.003 best
                                          Txn     Think
Key      RT      RT      Menu
                                       Weight  Time
Time     Delay   Fence   Delay
```

```
                    New Order          44.91
12.09       18.01       0.10      5.00       0.10
                    Payment            43.02
12.09        3.01       0.10      5.00       0.10
                    Delivery            4.02
5.07         2.01       0.10      5.00       0.10
                    Stock Level         4.02
5.07         2.01       0.10     20.00       0.10
                    Order Status        4.03
10.08        2.01       0.10      5.00       0.10

                    ExtraKick
                    FullSpeedKick
                                      Txn     Think
Key     RT      RT      Menu
                                      Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.92
12.03       18.01       0.10      5.00       0.10
                    Payment            43.01
12.03        3.01       0.10      5.00       0.10
                    Delivery            4.02
5.03         2.01       0.10      5.00       0.10
                    Stock Level         4.02
5.03         2.01       0.10     20.00       0.10
                    Order Status        4.03
10.03        2.01       0.10      5.00       0.10

                    ovd_11

                                      Txn     Think
Key     RT      RT      Menu
                                      Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.92
10.85       18.00       0.10      5.00       0.10
                    Payment            43.01
10.85        3.00       0.10      5.00       0.10
                    Delivery            4.02
4.55         2.00       0.10      5.00       0.10
                    Stock Level         4.03
4.55         2.00       0.10     20.00       0.10
                    Order Status        4.02
9.05         2.00       0.10      5.00       0.10

                    ovd_10

                                      Txn     Think
Key     RT      RT      Menu
                                      Weight  Time
Time    Delay   Fence   Delay
                    New Order          44.92
10.12       18.00       0.10      5.00       0.10
                    Payment            43.01
10.12        3.00       0.10      5.00       0.10
                    Delivery            4.02
4.24         2.00       0.10      5.00       0.10
                    Stock Level         4.03
4.24         2.00       0.10     20.00       0.10
                    Order Status        4.02
8.44         2.00       0.10      5.00       0.10
```

Appendix D:

60-Day Space

TPC-C 60 Day Space Requirements								
Warehouses	84,000				TpmC	**1,024,381**		
Table	Rows	Data KB	Index KB	Extra 5% KB	8hr Space	Total Space KB		dev\tpcc
Warehouse	84,000	**8,960**	**888**	492		10,340		10,340
District	840,000	**93,336**	**1,472**	4,740		99,548		99,548
Customer	2,520,000,000	**1,832,727,280**	**114,353,216**	97,354,025		2,044,434,521		2,044,434,521
History	2,651,101,849	**155,975,024**	**1,113,736**		27,774,892	157,088,760		184,863,652
New_order	767,480,282	**20,210,256**	**74,456**	1,014,236		21,298,948		21,298,948
Orders	2,654,341,122	**97,569,032**	**80,520,904**		18,718,690	178,089,936		196,808,626
Order_line	26,543,952,039	**1,812,068,536**	**7,798,192**		548,510,355	1,819,866,728		2,368,377,083
Item	100,000	**9,416**	**864**	514		10,794		10,794
Stock	8,400,000,000	**2,688,000,000**	**5,670,632**	134,683,532		2,828,354,164		2,828,354,164
								0
Total		6,606,661,840	209,534,360	233,057,539	595,003,938	7,049,253,739		7,644,257,677
	MB						files=	**162**
Dynamic Space	2,017,200	Sum of Data for Order, Orderline and History					size=	**9,086,720**
Static Space	4,866,837	Sum of Data+Index+5%-Dynamic Space					Total=	1,472,048,640
Free Space	na	Total Allocated Spac - (Dynamic + Static Space)					8K blocks	11,776,389,120
Daily Growth	393,596	(Dynamic Space/(W*62.5))*tpmc						OK
Daily Spread	-	(Free Space -1.5*Daily Growth) Zero Assumed						
60 Day Space MB	28,482,610	MB						
60 Day Space GB	27,815.05	GB						
Log Size	3,500,000.00	MB						
KB Per New Order	6.44	KB						
8 hr log MB	3,092,830	MB						
8 hr log GB	3,020.34	GB						
Space Usage	GB Needed	Disks Measured	GB Priced	Disk Size	Formatted Size			
60 Day Space DB	27,815	81	30,181.41	400GB	372.61			
		54	25,151.04	500GB	465.76			
Total DB			55,332.45					
8-hr log + mirror	6,041	48	6,554.40	149GB	136.55			
OS, Swap	3	2	138.00	72GB	69.00			
Total Storage	33,858.73	GB	62,024.85	GB				

tpmC	1,024,381									
	Data Before KB	Index Before KB	Data After KB	Index After KB	Data Grow KB	Index Grow KB	Total Grow KB	KB/New-Order	8-Hr Growth KB	8-Hr Growth MB
History	155,975,024	1,113,736	169,415,392	1,113,736	13,440,368	0	13,440,368	0.0565	27,774,892.37	27,123.92
Order	97,569,032	80,520,904	106,626,624	80,521,352	9,057,592	448	9,058,040	0.0381	18,718,690.30	18,279.97
Order-Line	1,812,068,536	7,798,192	2,077,494,624	7,798,192	265,426,088	0	265,426,088	1.1155	548,510,355.27	535,654.64
	sum(*) Before		sum(*) After		Num New-Order					
d_next_o_id	2,655,181,122		2,893,117,757		237,936,635					
	Before MB		After MB		Grow MB			KB/New-Order	8-Hr Growth MB	8-Hr Growth GB
Log	915,295.50		2,411,927.00		1,496,631.50			6.4410	3,092,830.41	3,020.34
								6,595.5874	bytes	
3,500,000.00	26.1513		68.9122							
Database tpcc log used (%)										

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

March 31, 2011

Hewlett-Packard
David Adams
20555 SH 249
Houston, TX 77070

Here is the information you requested regarding pricing for several Microsoft products to be used in conjunction with your TPC-C benchmark testing.

All pricing shown is in US Dollars ($).

Part Number	Description	Unit Price	Quantity	Price
810-03134	**SQL Server 2005 Enterprise Edition** *Per Processor License* *Discount Schedule: Open Program - Level C* *Unit Price reflects a 6% discount from the retail unit price of $24,999.*	$23,432	2	$46,864
P72-04217	**Windows Server 2008 R2 Enterprise Edition** *Server License with 25 CALs* *Discount Schedule: Open Program - Level C* *Unit Price reflects a 43% discount from the retail unit price of $3,999.*	$2,280	8	$18,240
127-00166	**Visual Studio 2008 Standard Edition** *Full License* *No Discount Applied*	$275	1	$275
N/A	**Microsoft Problem Resolution Services** *Professional Support* *(1 Incident).*	$259	1	$259

SQL Server 2005 Enterprise Edition, Windows Server 2008 R2 Enterprise Edition, and Visual Studio 2008 Standard Edition are currently orderable and available through Microsoft's normal distribution channels. A list of Microsoft's resellers can be found at the Microsoft Product Information Center at **http://www.microsoft.com/products/info/render.aspx?view=22&type=how**

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at $259 per call.

This quote is valid for the next 90 days.

Reference ID: TPCC_g3wOpiq6ZAsO7Lbitf7N9cl1uGYEbDe4_V1.0.0.

MICROLAND ELECTRONICS

Logout | My Account | Comments

HOME | PRODUCT | ORDER STATUS | RMA | SUPPORT | SERVICE | ABO

Shopping Cart

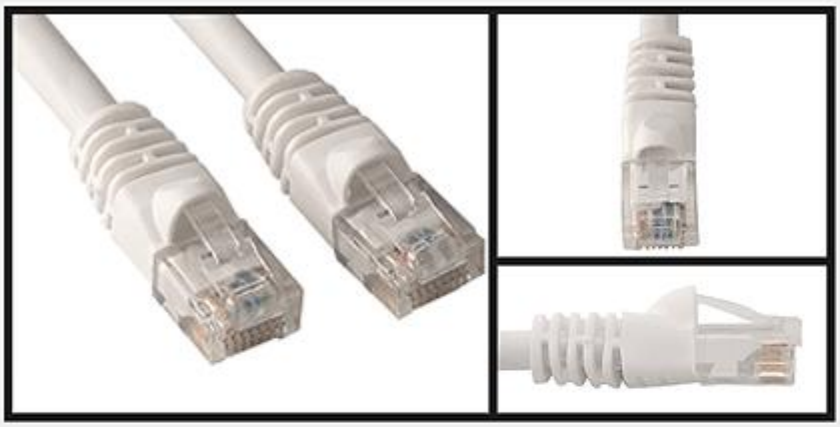
keep Shopping | Recalculate | Check

Part #	Description	Unit Price	Qty	Qty X Price	Delet
LSIS920016B	LSI HBA 9200-16e 16 External ports PCI-E RAID JBOD SAS RAID Controller, No Cable BOX SGL RoHS	$528.00	4	$2,112.00	
LSIS920080B	LSI HBA 9200-8e 8 External ports SAS Controller, No cable BOX SGL RoHS	$355.00	1	$355.00	

Product ID	Summary	Unit Price	Qty	Price
LSIS920016B	LSI HBA 9200-16e 16 External ports PCI-E RAID JBOD SAS RAID Controller, No Cable BOX SGL RoHS	$528.00	4	$2,112.00
LSIS920080B	LSI HBA 9200-8e 8 External ports SAS Controller, No cable BOX SGL RoHS	$355.00	1	$355.00

	Sub-Total:	$2,467.00
UPS Ground Service	Shipping & Handling Charge:	$67.00
	Tax:	$209.05
	Total Amount:	$2,743.05

7ft White Cat 6 Patch Cable, Molded
As low as $1.16



Quantity	Price
1 – 2	$1.70
3 – 4	$1.67
5 – 7	$1.63
8 – 11	$1.61
12 – 15	$1.59
16 – 19	$1.58
20 – 29	$1.57
30 – 39	$1.55
40 – 49	$1.54
50 – 99	$1.52
100 – 199	$1.34
200 – 299	$1.31
300 – 499	$1.27
500 – 999	$1.19
1000 +	$1.16

1 Purchase
ADD TO CART TO ESTIMATE SHIPPING

ShareThis New

Meets or exceeds the ANSI/TIA/EIA-568-B.2-1 standard for CAT 6 CMR, communication riser cable, and certified by UL, Un
on each end and boots to protect the tab of the RJ45 connector from being snagged. Packaged individually in labeled bags.

Part #: CB242-7W Tell a Friend

Condition: New
Mfg: Abergetty

Other great items you might enjoy:

7ft Yellow Cat 6 Patch Cable Molded
As low as $1.16

3ft Blue Cat 6 Patch Cable, Molded
As low as $0.84

5ft Yellow Cat 6 Patch Cable Molded
As low as $1.13

14ft Blue Cat 6 Patch Cable, Molded
As low as $2.38

Appendix F: Price Verification and Availability

The 400 GB SSD SAS drives will be available on June 20, 2011. All other hardware is currently available
HP Direct: 800-203-6748

For price verification before order date: e-mail hp.pricing.desk@hp.com