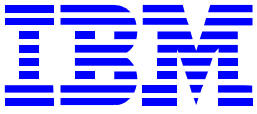
IBM Power 550

Using
Sybase ASE 15.0.3
and
Red Hat Enterprise Linux Advanced Platform 5 Update 1

TPC Benchmark™ C
Full Disclosure Report



Second Edition August 25, 2009

Special Notices

The following terms used in this publication are trademarks of **International Business Machines** Corporation in the United States and/or other countries:

IBM System p

IBM System x

IBM

The following terms used in this publication are trademarks of other companies as follows:

TPC Benchmark, TPC-C, and tpmC are trademarks of the Transaction Processing Performance Council

Sybase ASE is a registered trademark of Sybase

Microsoft Windows 2003 server and COM+ are registered trademarks of Microsoft Corporation

Linux is a registered trademark of Linus Torvalds

Red Hat and Red Hat Enterprise Linux are registered trademarks of Red Hat, Inc.

First Edition: June 16, 2008

Request for additional copies of this document should be sent to the following address:

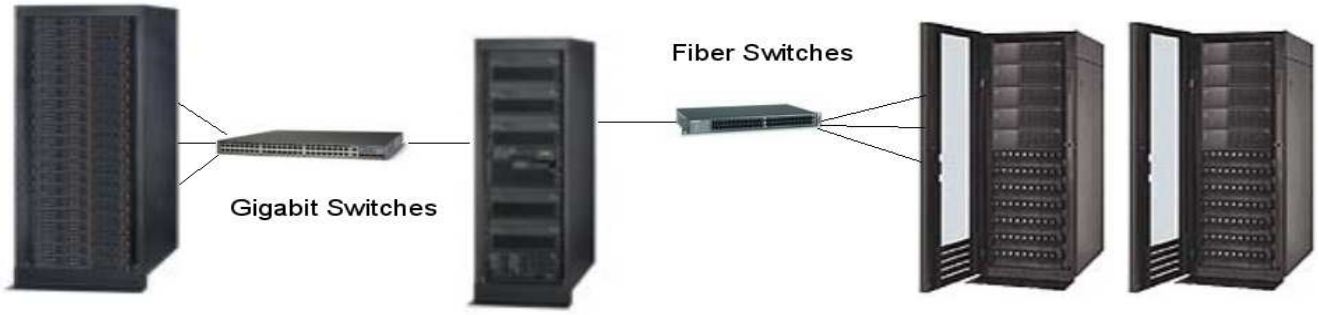
TPC Benchmark Administrator
IBM Commercial Performance
Mail Stop 9571
11501 Burnet Road
Austin, TX 78758
FAX Number (512) 838-1852

IBM	**IBM Power 550 Express**	**TPC-C Rev. 5.10** **Report Date: June 16, 2008** **Revision Date: August 25, 2009**

Total System Cost	TPC-C Throughput	Price/Performance	Availability Date
$702,996 USD	**276,383**	$2.55 USD	**December 16, 2008**

Database Processors/Cores/Threads	Database Manager	Operating System	Other Software	No. Users
2/4/8 4.2 GHz POWER6	Sybase ASE 15.0.3	RHEL 5.1	Microsoft Visual C++ Microsoft COM+	220,000

Priced



8 Clients
IBM® System x3350
1x Intel X3360 QC
2.83GHz
1GB Memory
73 GB Internal SAS Drive
2 Gb Ethernet

IBM® Power 550
2 x POWER6 4.2GHz
128GB Memory
4 x IBM Dual Port 4GB FC
2 x Integrated Gb Ethernet
1 x 10/100/1000 dual port Ethernet

Storage
11 x IBM® DS3400 Storage Servers
30x IBM® EXP3000 DiskEnclosures
488 x 73 GB 15K RPM SAS Drives
2 x SAN16B Fiber Switch

System Components	Each of the 8 Clients		Server	
	Quantity	**Description**	**Quantity**	**Description**
Processors/Cores/ Threads	1/4/4	2.83GHz 12MB L2 Xeon X3360 Quad Core	2/4/8	4.2 GHz POWER6
Memory	1	1 GB	16	8 GB
Disk Controllers	1	SAS	1 4 10 1	Integrated SAS 4Gb dual-port FC Adapters IBM DS3400 Controllers for Data IBM DS3400 Controller for Log
Disk Drives	1	73 GB	490	73 GB 15K RPM SAS
Total Storage		584 GB		35,624 GB
Terminals	1	System Console	1	System Console

IBM Corporation	IBM Power 550 Express	TPC-C Rev. 5.10
	Sybase ASE 15.0.3	Report Date: June 16, 2008 Revision Date: August 25, 2009

Server Hardware	Part Number	Price Source	Unit Price	Qty	Ext. Price	3 -Yr. Maint
IBM Power 550 Express	8204-E8A5	1	5,509	1	5,509	
Op Panel Cable for Rack-mount Drawer w/3.5 DASD	1877	1	6	1	6	
73.4GB 15K RPM SAS Disk Drive	3646	1	498	2	996	
16384MB (2x8192MB) RDIMMs, 400 MHz, 1Gb Stacked DRAM	4524	1	13,926	8	111,408	
2-core 4.2 GHz POWER6 Processor Card	4966	1	11,828	2	23,656	
One Processor Activation for Processor Feature #4966	4986	1	7,573	2	15,146	
Dual-Port 1Gb Integrated Virtual Ethernet Daughter Card	5623	1	301	1	301	
IDE Slimline DVD-ROM Drive	5756	1	208	1	208	
4 Gb Dual-Port Fibre Channel PCI-X 2.0 DDR	5759	1	2,499	2	4,998	
2-Port 10/100/1000 Base-TX Ethernet PCI Express Adapter	5767	1	528	1	528	
4 Gigabit PCI Express Dual Port Fibre Channel Adapter	5774	1	2,499	2	4,998	
Power Cable -- Drawer to IBM PDU, 14-foot	6458	1	14	1	14	
IBM/OEM Rack-Mount Drawer Rail Kit	7146	1	300	1	300	
IBM Rack-mount Drawer Bezel and Hardware	7360	1	450	1	450	
Power Supply, 1700 Watt AC, Hot-swap, Base	7707	1	699	1	699	
DASD/Media Backplane for 3.5 DASD/DVD/Tape	8341	1	340	1	340	
Deskside Hardware Management Console	7042-C06	1	1,830	1	1,830	
T117 Color Display	3645	1	875	1	875	
IBM Quiet Touch USB Keyboard	5951	1	104	1	104	
24x7x4 3 Year Maintenance		1				7540
				Subtotal	172,366	7,540
Server Storage						
IBM System Storage EXP3000	1727-HC1	1	3,199	30	95,970	
IBM 1M SAS cable	39R6529	1	119	60	7,140	
IBM DS3000 Environmental Services Module (ESM)	39R6515	1	999	30	29,970	
IBM System Storage DS3400 Express	1726-42X	1	9,295	11	102,245	
IBM Hot-Swap 3.5 inch 73.4GB 15K SAS HDD	43W7523	1	309	488	150,792	
IBM TotalStorage SAN16B-2 Express Model	200516B	1	4,120	2	8,240	
IBM 4 Gbps SW SFP Transceiver	22R4902	1	150	21	3,150	
IBM 5m LC-LC Fiber Cable	39M5697	1	129	21	2,709	
16B Four Port Activation	22R4901	1	1,670	4	6,680	
IBM S2 42U Standard Rack	93074RX	1	1,489	2	2,978	
DS3000 Linux on Power Host License	44W2101	1	1,000	11	11,000	
ServicePac for 3-Year 24x7x4 Support (DS3400)	44J8073	1	1,300	11		14,300
ServicePac for 3-Year 24x7x4 Support (EXP3000)	41L2768	1	760	30		22,800
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	1	300	2		600
				Subtotal	420,874	37,700
Server Software						
Sybase ASE 15.0.3 Enterprise Edition for Linux on POWER (includes a 25% adjustment for multi-core)	14023	2	33,203	4	132,810	
Sybase SupportNow Standard Plan (includes a 25% adjustment for multi-core)	98480	2	29,217	3		87,651
Red Hat Enterprise Linux, Premium Subscription		3	1,299	3	3,897	
				Subtotal	136,707	87,651
Client Hardware						
IBM System x3350 (Quad-Core Intel Xeon x3360 2.83GHz, (2x1GB RAM)	419284U	1	1,869	8	14,952	
73.4GB 15K 2.5" SAS Hot-Swap Drive	43X0837	1	369	8	2,952	
ServicePac for 3-Year 24x7x4 Support (x3350)	51J9100	1	700	8		5,600
				Subtotal	17,904	5,600
Third Party Hardware/Software						
Microsoft Windows Server 2003 Web Edition with COM+	P70-00275	4	395	8	3,160	
Visual Studio Standard 2005	127-00012	5	250	1	250	
Microsoft Problem Resolution Services		5	245	1		245
D-LINK DGS-1024D 24-Port 10/100/1000 Switch (2 spares)	DGS1024D	6	174	3	522	
UPS - powercom KIN-1500AP 1500VA 900Watts Item # N82E16842106115 (2 spares)		7	150	3	450	
				Subtotal	4,382	245
				Total	752,233	138,736
					Total IBM Discount *	-$154,904
					Sybase 15% Discount	-$33,069
					Three-Year Cost of Ownership USD:	$702,996
Pricing Sources: 1) IBM 2) Sybase 3) Redhat 4) cdw.com 5) Microsoft					**tpmC:**	276,383
6) compuplus.com 7) newegg.com					**$ USD/tpmC:**	$2.55

* Discounts are based on US list prices for similar quantities & configurations including pre-payment for maintenance. The discount of 23.4% applies to the totality of all items with prices source "1".

Audited by: Francois Raab Info Sizing (www.infosizing.com)

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

Numerical Quantities Summary for the IBM Power 550

MQTH, computed Maximum Qualified Throughput: 276,383 tpmC

Response Times (in seconds)	90th %	Average	Maximum
New Order	0.700	0.346	6.000
Payment	0.680	0.330	5.078
Order-Status	0.740	0.394	6.094
Delivery (interactive)	0.120	0.118	4.172
Delivery (deferred)	1.93	0.84	4.53
Stock-Level	0.720	0.366	4.313
Menu	0.111	0.117	5.156

Response time delay added for emulated components was 0.1 seconds

Transaction Mix, in percent of total transactions	Percent
New Order	44.979%
Payment	43.000%
Order-Status	4.007%
Delivery	4.006%
Stock-Level	4.007%

Keying/Think Times (in seconds)	Min.	Average	Max.
New Order	18.00/0.00	18.00/12.035	18.016/120.313
Payment	3.00/0.00	3.00/12.035	3.016/120.313
Order-Status	2.00/0.00	2.00/10.035	2.016/100.313
Delivery	2.00/0.00	2.00/5.036	2.00/50.313
Stock-Level	2.00/0.00	2.00/5.036	2.00/50.313

Test Duration

Ramp-up Time	1 hour 36 minutes
Measurement interval	2 hours 28 minutes
Transactions during measurement interval (all types)	90,940,891

Checkpoints

Number of checkpoints	4
Checkpoint interval	29:50

Table of Contents

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification Revision 5.10 dated June, 2008, for measurements on the IBM Power 550. The software used on the IBM Power 550 includes Red Hat Enterprise Linux Advanced Platform 5 Update 1 operating system and Sybase ASE 15.0.3 data server. Microsoft COM+ is used as the transaction manager.

IBM Power 550

Company Name	System Name	Data Base Software	Operating System Software
IBM Corporation	IBM Power 550	Sybase ASE 15.0.3	Red Hat Enterprise Linux Advanced Platform 5 Update 1

Total System Cost	TPC-C Throughput	Price/Performance
• Hardware • Software • 3 Years Maintenance	Sustained maximum throughput of system running TPC-C expressed in transactions per minute	Total system cost/tpmC
$702,996 USD	276,383	$2.55 USD

Preface

TPC Benchmark™ C Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on August 13, 1992 and updated with revision 5.10 in June 2008.

This is the full disclosure report for benchmark testing of the IBM Power 550 and Sybase ASE 15.0.3 according to the TPC Benchmark™ C Standard Specification.

TPC Benchmark™ C exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing a mixture of read-only and update intensive transactions. This is a complex OLTP application environment exercising a breadth of system components associated by such environments 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
- Data bases consisting of many tables with a wide variety of sizes, attributes, and relationships
- Contention on data access and update

This benchmark defines four on-line transactions and one deferred transaction, intended to emulate functions that are common to many OLTP applications. However, this benchmark does not reflect the entire range of OLTP requirements. 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 benchmarks when critical capacity planning and/or product evaluation decisions are contemplated.

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.

0 General Items

0.1. Application Code Disclosure

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

Appendix A contains the application code for the five TPC Benchmark™ C transactions.

0.2. Benchmark Sponsor

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

This benchmark was sponsored by **International Business Machines Corporation.**

0.3. 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 but not limited to:

- *Data Base tuning options*
- *Recovery/commit options*
- *Consistency/locking options*
- *Operating system and application configuration parameters.*

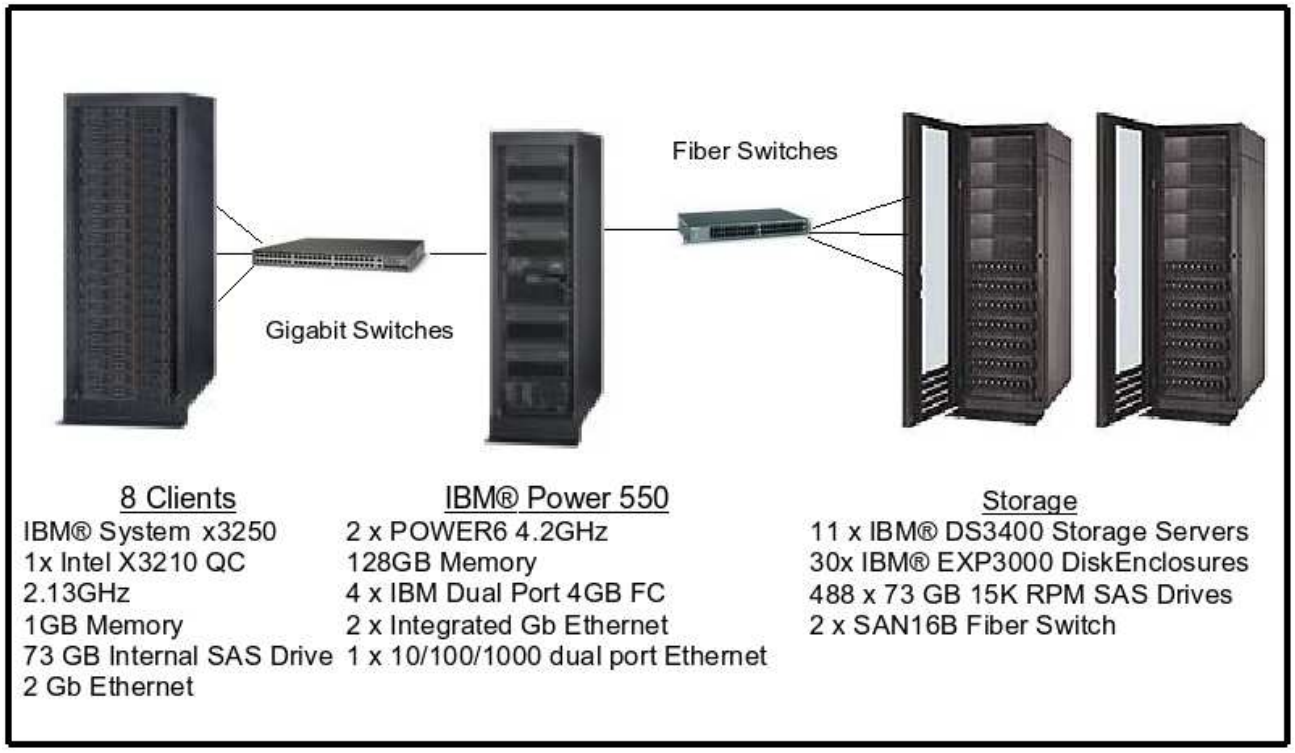
Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

0.4. Configuration Diagrams

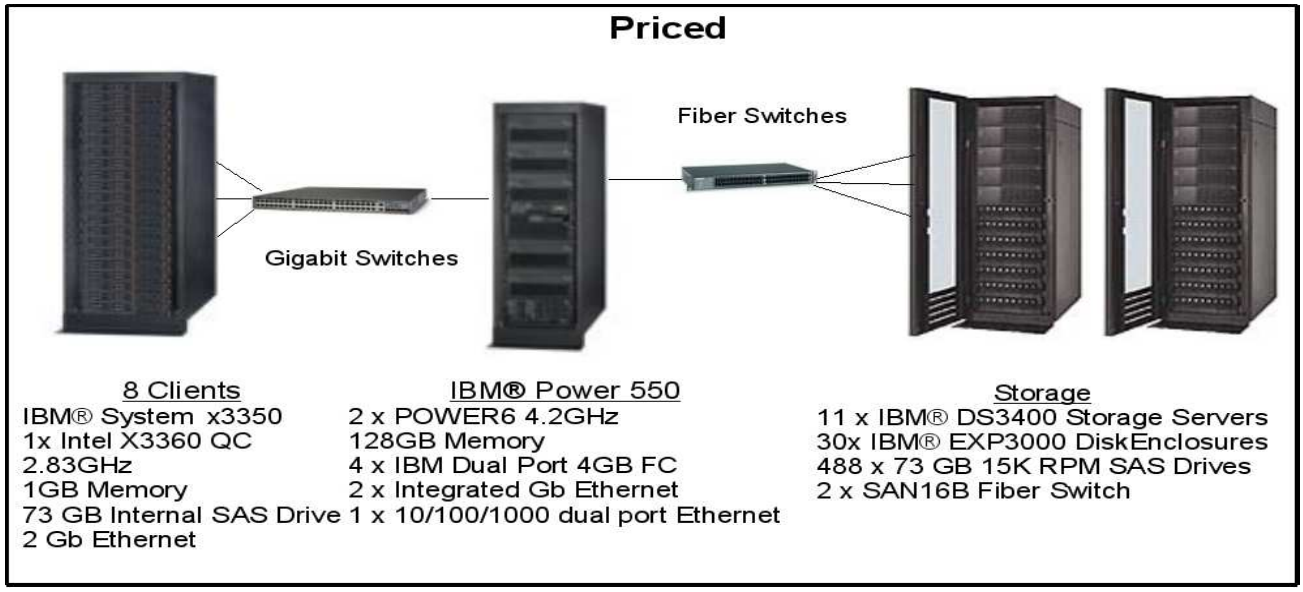
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *Number and type of processors*
- *Size of allocated memory, and any specific mapping/partitioning of memory unique to the test*
- *Number and type of disk units (and controllers, if applicable)*
- *Number of channels or bus connections to disk units, including the protocol type*
- *Number of LAN (e.g. Ethernet) connections, including routers, work stations, terminals, etc, that were physically used in the test or are incorporated into the pricing structure (see Clause 8.1.8)*
- *Type and run-time execution location of software components (e.g. DBMS, client processes, transaction monitors, software drivers, etc)*

IBM Power 550 Benchmark Configuration



IBM Power 550 Priced Configuration



Eight 4192-84U x3350 clients were priced for the 8 4365-8BU x3250 clients used in the benchmark. The priced systems have a faster processor (2.83GHz vs 2.13GHz) with a larger L2 cache (12MB vs. 8MB) and a faster memory bus (1333MHz vs.1066MHz). The auditor reviewed the specifications of the 2 machines and certified that the pricing upgrade was compliant with the TPC-C specification.

1 Clause 1: Logical Data Base Design Related Items

1.1. Table Definitions

Listings must be provided for all table definition statements and all other statements used to setup the data base.

Appendix C contains the table definitions and the database load programs used to build the data base.

1.2. Database Organization

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

Physical space was allocated to Sybase on the server disks according to the details provided in Appendix C.

1.3. Insert and/or 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 restriction in the SUT data base implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows.

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

The insert and delete functions were verified by the auditor. In addition, the auditor verified that the primary key for each database table could be updated outside the range of its initial partition.

1.4. Horizontal or Vertical Partitioning

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

Horizontal partitioning was used on the History table using functionality provided by Sybase Adaptive Server. For details please see appendix C.

2 Clause 2: Transaction & Terminal Profiles Related Items

2.1. Verification for the Random Number Generator

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

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 were found.

2.2. Input/Output Screens

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

The screen layouts are now presented in HTML 1.0 web pages. Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3 of the TPC-C specifications were used as guidelines for html character placement.

2.3. Priced Terminal Features

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 emulated workstations, IBM System x3250 systems, are commercially available and support all of the requirements in Clause 2.2.2.4.

2.4. Presentation Managers

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

The workstations did not involve screen presentations, message bundling or local storage of TPC-C rows. All screen processing was handled by the client system. All data manipulation was handled by the server system.

2.5. Home and Remote Order-lines

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

Table 2-1 shows the percentage of home and remote transactions that occurred during the measurement period for the New-Order transactions.

2.6. New-Order Rollback Transactions

The percentage of New-Order transactions that were rolled back as a result of an illegal item number must be disclosed.

Table 2-1 shows the percentage of New-Order transactions that were rolled back due to an illegal item being entered.

2.7. Number of Items per Order

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

Table 2-1 show the average number of items ordered per New-Order transaction.

2.8. Home and Remote Payment Transactions

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

Table 2-1 shows the percentage of home and remote transactions that occurred during the measurement period for the Payment transactions.

2.9. Non-Primary Key Transactions

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

Table 2-1 shows the percentage of non-primary key accesses to the data base by the Payment and Order-Status transactions.

2.10. Skipped Delivery Transactions

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

Table 2-1 shows the percentage of Delivery transactions missed due to a shortage of supply of rows in the NEW-ORDER table.

2.11. Mix of Transaction Types

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

Table 2-1 shows the mix percentage for each of the transaction types executed by the SUT.

New Order	**IBM Power 550**
Percentage of Home order lines	99.00%
Percentage of Remote order lines	1.00%
Percentage of Rolled Back Transactions	1.00%
Average Number of Items per order	10
Payment	
Percentage of Home transactions	84.99%
Percentage of Remote transactions	15.01%
Non-Primary Key Access	
Percentage of Payment using C_LAST	59.999%
Percentage of Order-Status using C_LAST	59.979%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.979%
Payment	43.000%
Order-Status	4.007%
Delivery	4.006%
Stock-Level	4.007%

Table 2-1: Numerical Quantities for Transaction and Terminal Profiles

2.12. Queuing Mechanism of Delivery

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

The Delivery transaction was submitted to an ISAPI queue that is separate from the COM+ queue that the other transactions used. This queue is serviced by a variable amount of threads that are separate from the worker threads inside the web server. Web server threads are able to complete the on-line part of the Delivery transaction and immediately return successful queuing responses to the drivers. The threads servicing the queue are responsible for completing the deferred part of the transaction asynchronously.

3 Clause 3: Transaction and System Properties

The results of the ACID test must be disclosed along with a description of how the ACID requirements were met.

All ACID tests were conducted according to specification.

3.1. Atomicity Requirements

The system under test must guarantee that data base 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.

3.1.1. Atomicity of Completed Transaction

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

The following steps were performed to verify the Atomicity of completed transactions.

1. The balance, BALANCE_1, was retrieved from the CUSTOMER table for a random Customer, District and Warehouse combination.
2. The Payment transaction was executed and committed for the Customer, District, and Warehouse combination used in step 1.
3. The balance, BALANCE_2, was retrieved again for the Customer, District, and Warehouse combination used in step 1 and step 2. It was verified that BALANCE_1 was greater than BALANCE_2 by the amount of the Payment transaction.

3.1.2. Atomicity of Aborted Transactions

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

The following steps were performed to verify the Atomicity of the aborted Payment transaction:

1. The Payment application code was implemented with a Perl script that allowed the transaction to be rolled back rather than committed.
2. The balance, BALANCE_3, was retrieved from the Customer table for the same Customer, District, and Warehouse combination used in the completed Payment transaction Atomicity test.
3. The Payment transaction was executed for the Customer, District and Warehouse used in step 2. Rather than commit the transaction, the transaction was rolled back.
4. The balance, BALANCE_4 was retrieved again for the Customer, District, and Warehouse combination used in step 2. It was verified that BALANCE_4 was equal to BALANCE_3, demonstrating that there were no remaining effects of the rolled back Payment transaction.

3.2. Consistency Requirements

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

Verify that the data base is initially consistent by verifying that it meets the consistency conditions defined in Clauses 3.3.2.1 to 3.3.2.4. Describe the steps used to do this in sufficient detail so that the steps are independently repeatable.

The specification defines 12 consistency conditions of which the following four are required to be explicitly demonstrated:

1. The sum of balances (d_ytd) for all Districts within a specific Warehouse is equal to the balance (w_ytd) of that Warehouse.
2. For each District within a Warehouse, the next available Order ID (d_next_o_id) minus one is equal to the most recent Order ID [max(o_id)] for the Order table associated with the preceeding District and Warehouse.

Additionally, that same relationship exists for the most recent Order ID [max(o_id)] for the New Order table associated with the same District and Warehouse. Those relationships can be illustrated as follows:

d_next_o_id – 1 = max(o_id) = max(no_o_id)

where (d_w_id = o_w_id = no_w_id) and (d_id = o_d_id = no_d_id)

3. For each District within a Warehouse, the value of the most recent Order ID [max(no_o_id)] minus the first Order ID [min(no_o_id)] plus one, for the New Order table associated with the District and Warehouse equals the number of rows in that New Order table. That relationship can be illustrated as follows:

   max(no_o_id) – min(no_o_id) + 1 = number of rows in New Order for the Warehouse/District

4. For each District within a Warehouse, the sum of Order Line counts [sum(o_ol_cnt)] for the Order table associated with the District equals the number of rows in the Order Line table associated with the same District. That relationship can be illustrated as follows:

   sum(o_ol_cnt) = number of rows in the Order Line table for the Warehouse/District

An RTE driven run was executed against a freshly loaded database. After the run the 4 consistency conditions defined above were tested using a script to issue queries to the database. All queries showed that the database was still in a consistent state.

3.3. Isolation Requirements

Operations of concurrent data base transactions must yield results which are indistinguishable from the results which would be obtained by forcing each transaction to be serially executed to completion in some order.

The benchmark specification defines nine tests to demonstrate the property of transaction isolation. The tests, described in Clauses 3.4.2.1 – 3.4.2.9, were all successfully executed using a series of scripts. Case A was observed during the execution of Isolation Tests 7-9.

3.4. Durability Requirements

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

- *Permanent irrecoverable failure of any single durable medium containing TPC-C database tables or recovery log data (this test includes failure of all or part of memory)*
- *Instantaneous interruption (system crash/system hang) in processing that requires system reboot to recover*
- *Failure of all or part of memory (loss of contents)*

Failure of Durable Medium containing Recover Log

This test was performed using 2800 warehouses.

The following steps were successfully performed:

1. The current count of the total number of orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving SUM_1.
2. A load test was started and allowed to run for over 10 minutes.
3. A single disk was removed from the RAID5 log device.
4. The RTE was allowed to run for 5 additinal minutes and then stopped.
5. The RTE run was terminated.
6. The database was shutdown.
7. The database was restarted and no database recovery was required.
8. Step 1 was performed returning the value for SUM_2. It was verified that SUM_2 - SUM1 was the same value as the number of RTE new orders - rollbacks.
9. Consistency condition 3 was verified.

Loss of Redundant Log Controller and Mirrored Cache

This test was performed using 2800 warehouses.

The following steps were successfully performed:

1. The current count of the total number of orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving SUM_1.
2. A load test was started and allowed to run for over 10 minutes.
3. One side (B) of the log controller was removed from the storage controller.
4. The battery was removed from the controller for 5 minutes.
5. The battery was reinserted into the controller and the controller was reinserted into the subsystem.
6. The RTE run was then terminated.
7. The database was shutdown and restarted. The database showed that no recovery was required.
8. Step 1 was performed returning the value for SUM_2. It was verified that SUM_2 - SUM1 was the same value as the number of RTE new orders - rollbacks.
9. Consistency condition 3 was verified.

Failure of Durable Medium containing TPC-C database tables

This test was performed using 2800 warehouses.

The following steps were successfully performed:

1. The current count of the total number of orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving SUM_1.
2. A backup (dump database tpcc) of the TPC-C database tables was performed.
3. A load test was started and allowed to run for over 10 minutes.
4. A disk containing the TPC-C datafiles was removed from the system, causing the database to fail. The transactions from the log were then dumped.
5. The RTE was then terminated.
6. The disk was placed back in service.
7. The disk was restored with original partition data.
8. The database was started and an attempt to recover the database failed.
9. The database was dropped and recreated for load.
10. The TPC-C backup data was loaded into the database (load database tpcc).
11. The transactions from the log were then applied to the database (load tran tpcc).
12. Step 1 was performed returning the value for SUM_2. It was verified that SUM_2 - SUM1 was the same value as the number of RTE new orders - rollbacks.
13. Consistency condition 3 was verified.

Instantaneous Interruption, Memory Failure, and Loss of Power:

This test was conducted on a fully-scaled database. The following steps were successfully performed:

The following steps were successfully performed:

1. The current count of the total number of orders was determined by the sum of D_NEXT_O_ID of all rows in the DISTRICT table giving SUM_1.
2. A full load test was started and allowed to run for over 10 minutes.
3. The system was subsequently powered off (power chords removed), which removed power from all system components, including memory.
4. The RTE was terminated.
5. The system was powered back on and the Sybase database was allowed to recover.
6. Step 1 was performed returning the value for SUM_2. It was verified that SUM_2 was greater than SUM_1 plus the completed New_Order transactions recorded by the RTE. The additional transactions found in the database were attributed to in-flight activity at the time of the failure.
7. Consistency condition 3 was verified.

4 Clause 4: Scaling and Data Base Population Related Items

4.1. Cardinality of Tables

The cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run, must be disclosed.

Table 4-1 portrays the TPC Benchmark™C defined tables and the number of rows for each table as they were built initially.

All tables are based on 22,000 warehouses, the number of active warehouses during the benchmark.

Table Name	Number of Rows
Warehouse	22,000
District	220,000
Customer	660,000,000
History	660,000,000
Order	666,000,000
New Order	198,000,000
Order Line	6,600,000,000
Stock	2,200,000,000
Item	100,000

Table 4-1: Initial Cardinality of Tables

4.2. Distribution of Tables and Logs

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

There is one Logical Disk (LD) for the logs.

There is one storage adapter for the log LD.

The log LD is configured as a RAID5 (7+p) disk array, with 8 physical disks. Each physical disk has a capacity of 73.4 GB. The total Raid 5 capacity available is 475 GB.

There are 20 Logical Disks (LD) for the tables.

There are 3 dual-ported storage adapters for the tables. A single fiber switch is used to connect all ports to the LDs.

Each LD is configured as a RAID0 disk array, with 24 physical disks.

There is a total of 480 data disks and each physical disk has a capacity of 73.4 GB drives.

Each LD contains 7 partitions. The partitions are laid out on an LD as follows:

Partition#	1K Blocks	Segment/Table Usage
1	6032376	Orders, New Orders, History
2	47319457	Stock
3	38716650	Customer
4	Extended partition	
5	2160711	Customer Index
6	32266521	Order Lines
		District table
7	1453851	Sybase Master
		Warehouse
		Item
8, 9	255963613	Not used for data tables

Segment/Tables are laid out to use LDs as follows:

Warehouse	1 table using 1 partition on 1 LD
District	1 table using LVM on 20 LDs
Item	1 table using 1 partition on 1 LD
Stock	1 table using 1 partition on 20 LDs
Customer	1 table using 1 partitions on 20 LDs
Customer Index	1 segment using 1 partition on 20 LDs
History	1 segment using 1 partition on 20 LDs
Orders	1 segment using 1 partition on 20 LDs
Order Line	1 table using LVM on 20 LDS
New Orders	1 segment using 1 partition on 20 LDS

4.3. Data Base Model Implemented

A statement must be provided that describes the data base model implemented by the DBMS used.

The database manager used for this testing was Sybase ASE 15.0.3. Sybase is a relational DBMS. Sybase remote stored procedures and embedded SQL statements were used. The Sybase stored procedures were invoked via SQL CALL statements. Both the client application and stored procedures were written in embedded C code.

4.4. Partitions/Replications Mapping

The mapping of data base partitions/replications must be explicitly described.

Horizontal partitioning was used on the History table using functionality provided by Sybase Adaptive Server. No tables were replicated in this benchmark test.

4.5. 60-Day Space Calculations

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

Note : Numbers are in KBytes unless otherwise specified

Warehouses	22000	**tpmC**	276383	**tpmC/W**	12.56
#wh used in test	22000				

Table	**Rows**	**Data**	**Index**	**5% Space**	**8H Space**	**Total Space**
Warehouse	22,000	88,008	4	4,401		92,413
District	220,000	880,008	4	44,001		924,013
Item	100,000	9,308	4	186		9,498
New-order	198,000,000	3,168,004	9,436		880,000	4,057,440
History	660,000,000	36,962,000	0		6,053,438	43,015,438
Orders	660,000,000	18,897,516	56,248		3,104,146	22,057,910
Customer	660,000,000	484,000,008	0	9,680,000		493,680,008
Cust index		0	30,379,756	607,595		30,987,351
Order-line	6,600,000,000	406,326,804	1,310,744		66,760,692	474,398,240
Stock	2,200,000,000	677,292,312	4	13,545,846		690,838,162
Totals		**1,627,623,968**	**31,756,200**	**23,882,029**	**76,798,276**	**1,760,060,473**

Segment	**LogDev Cnt.**	**Seg. Size**	**Needed**	**Overhead**	**Not Needed**
warehouse	1	131,072	93,337	933	37,735
district	1	1,290,240	933,253	9,333	356,987
item	1	16,384	9,593	96	6,791
order			22,278,489	222,785	
new_order			4,098,014	40,980	
history			43,445,593	434,456	
order +new_order +history	20	114,688,000	69,822,096	698,221	44,865,904
customer	20	737,280,000	498,616,808	4,986,168	238,663,192
customer index	20	40,960,000	31,297,225	312,972	9,662,775
order_line	20	614,400,000	479,142,222	4,791,422	135,257,778
stock	20	901,120,000	697,746,544	6,977,465	203,373,456
Totals		2,409,885,696	1,777,661,078	17,776,611	632,224,618

Dynamic space	447,858,544	Sum of Data for Order, Order-Line and History (excluding free extents)
Static space	1,253,180,264	Data + Index + 5% Space + Overhead - Dynamic space
Free space	76,622,270	Total Seg. Size - Dynamic Space - Static Space - Not Needed
Daily growth	90,022,173	(Dynamic space/W * 62.5)* tpmC
Daily spread	(58,410,990)	Free space - 1.5 * Daily growth (zero if negative)
60 day (KB)	6,654,510,649	Static space + 60 (daily growth + daily spread)
60 day (GB)	6346.24	Excludes OS, Paging and RDBMS Logs
config data(GB)	32575.55	
Log per N-O txn	0.92	Number of 4K blocks per New-Order transaction based on average log
8 Hour Log (GB)	465.59	consumption from test run
config log size (GB)	474.66	

5 Clause 5: Performance Metrics and Response Time Related Items

5.1. 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 5-1 lists the response times and the ninetieth percentiles for each of the transaction types for the measured system.

5.2. Keying and Think Times

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

Table 5-1 lists the TPC-C keying and think times for the measured system.

Response Times	New Order	Payment	Order Status	Delivery (int./def.)	Stock Level	Menus
90 %	0.700	0.680	0.740	0.120/1.93	0.720	0.111
Average	0.346	0.330	0.394	0.118/0.84	0.366	0.117
Maximum	6.000	5.078	6.094	4.172/4.53	4.313	5.156
Think Times						
Minimum	0	0	0	0	0	N/A
Average	12.035	12.035	10.035	5.036	5.036	N/A
Maximum	120.313	120.313	100.313	50.313	50.313	N/A
Keying Times						
Minimum	18.00	3.00	2.00	2.00	2.00	N/A
Average	18.00	3.00	2.00	2.00	2.00	N/A
Maximum	18.016	3.016	2.016	2.00	2.000	N/A

Table 5-1: Think and Keying Times

5.3. Response Time Frequency Distribution

Response time frequency distribution curves must be reported for each transaction type.

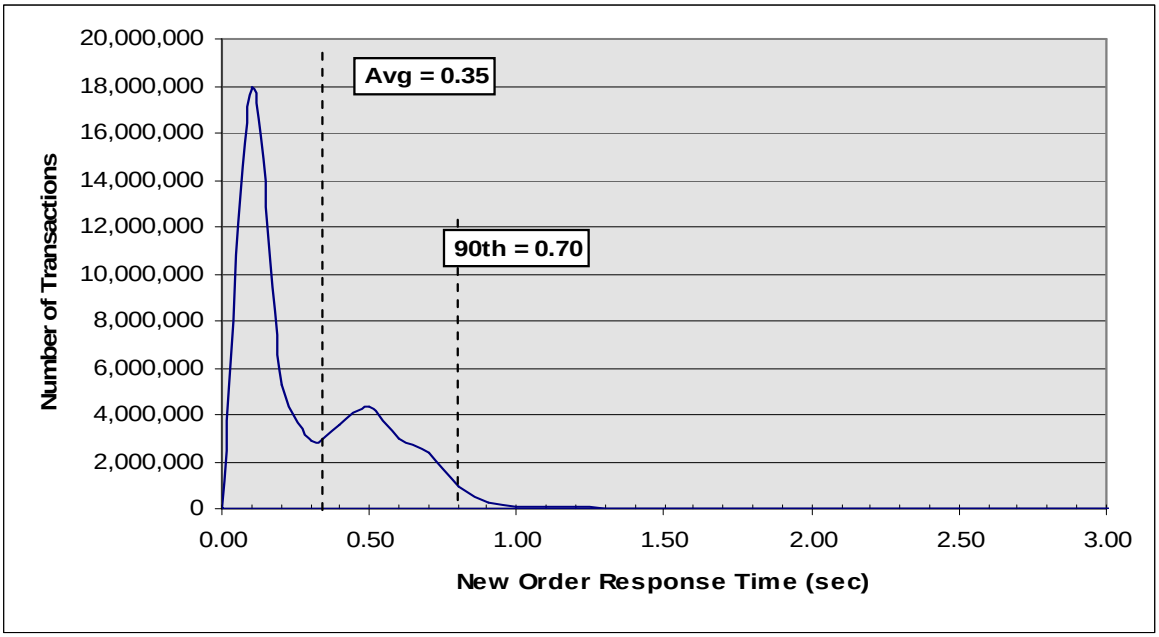


Figure 5-1: New-Order Response Time Distribution

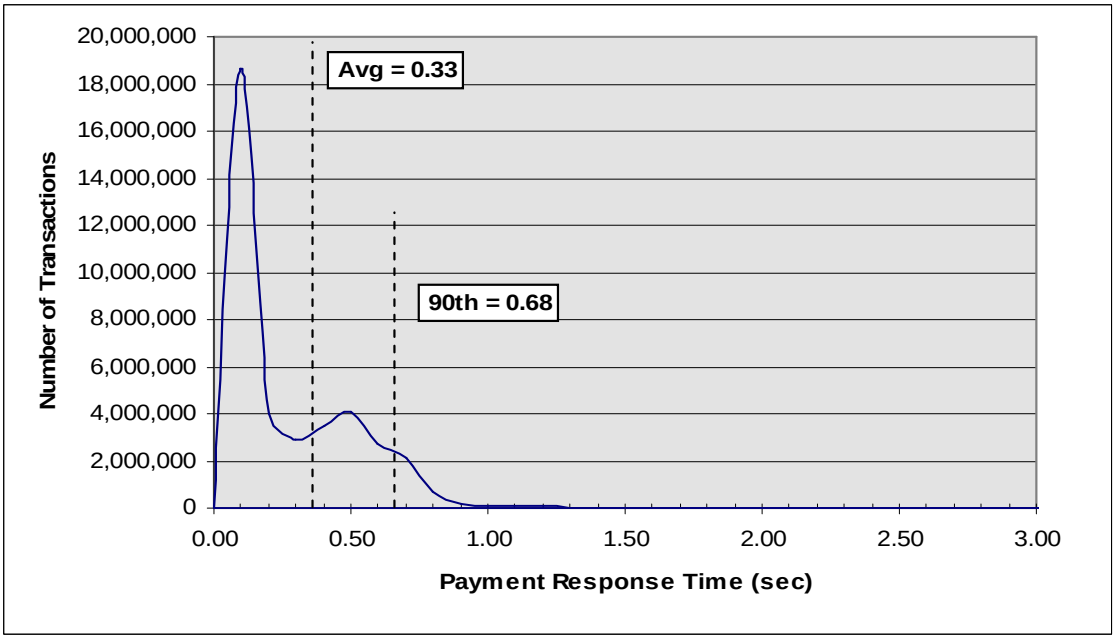


Figure 5-2: Payment Response Time Distribution

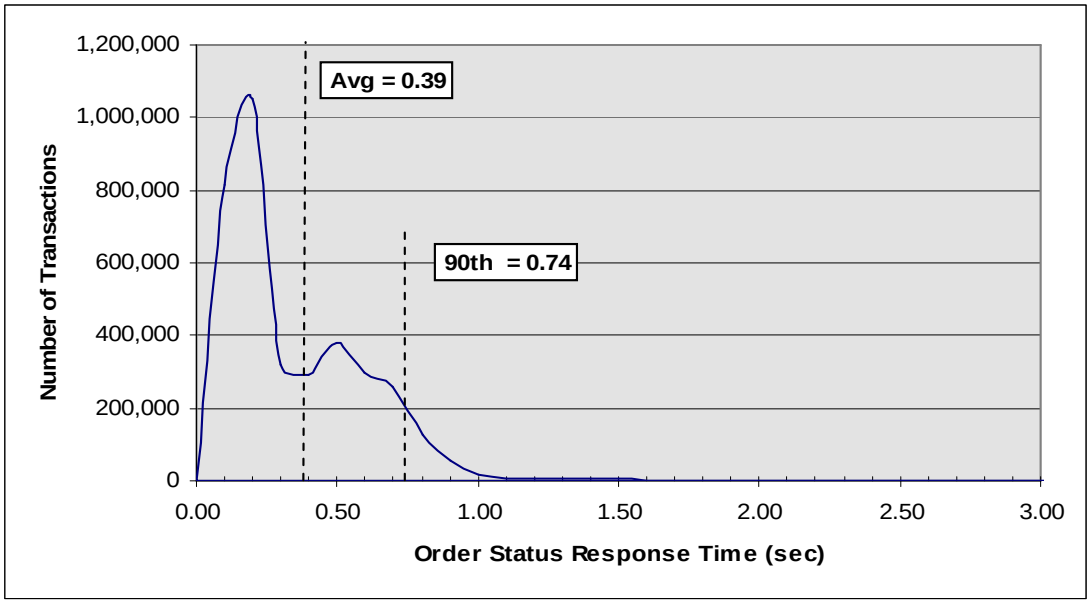


Figure 5-3: Order-Status Response Time Distribution

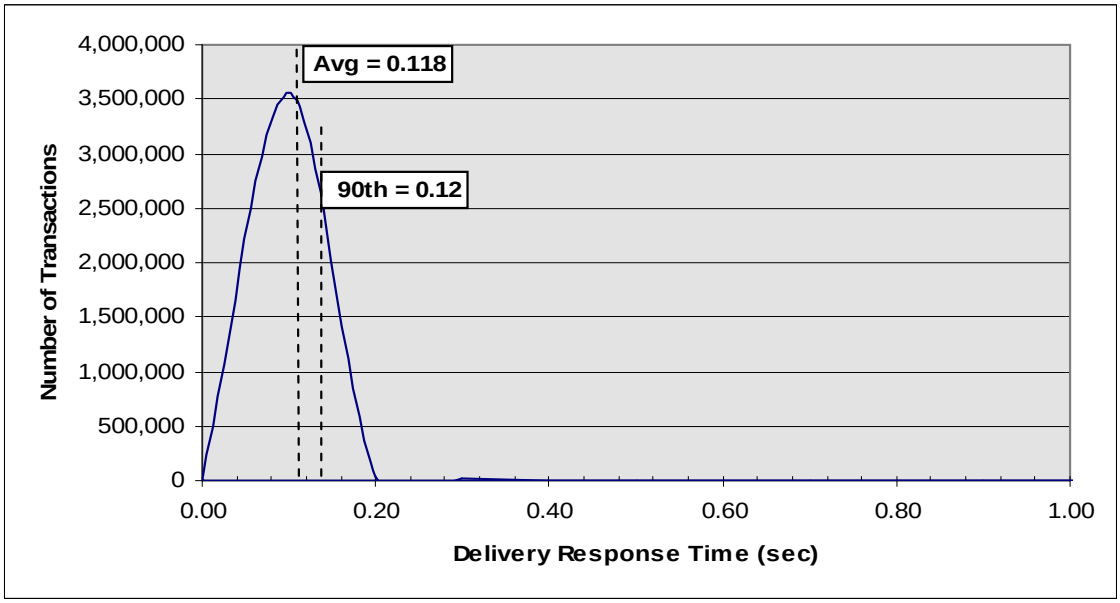


Figure 5-4: Delivery (Interactive) Response Time Distribution

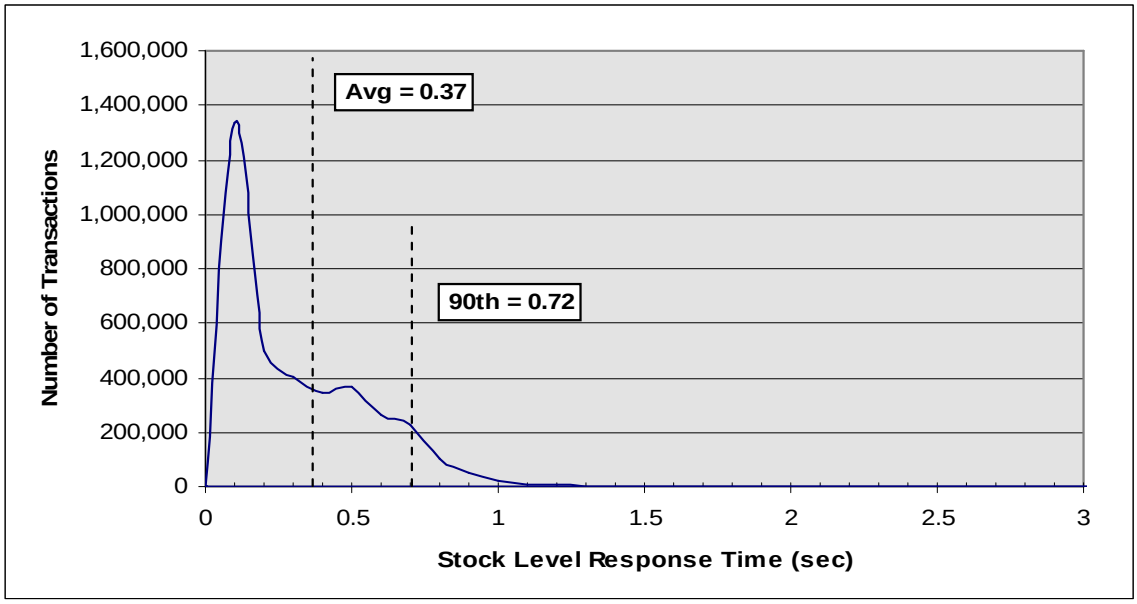


Figure 5-5: Stock Level Response Time Distribution

5.4. Performance Curve for Response Time versus Throughput

The performance curve for response times versus throughput must be reported for the New-Order transaction.

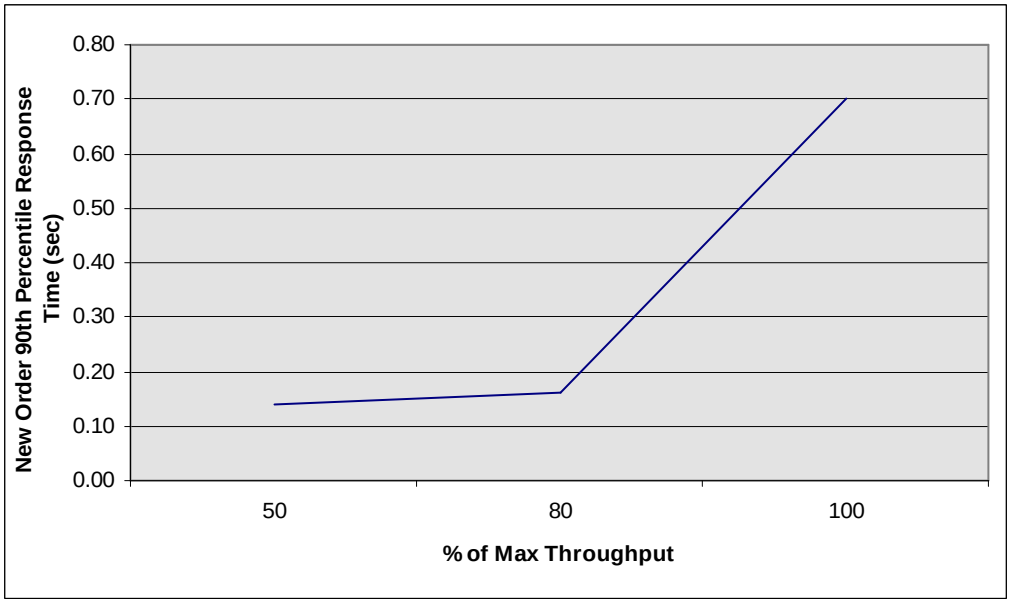


Figure 5-6: New-Order Response Time vs. Throughput

5.5. Think Time Frequency Distribution

A graph of the think time frequency distribution must be reported for the New-Order transaction.

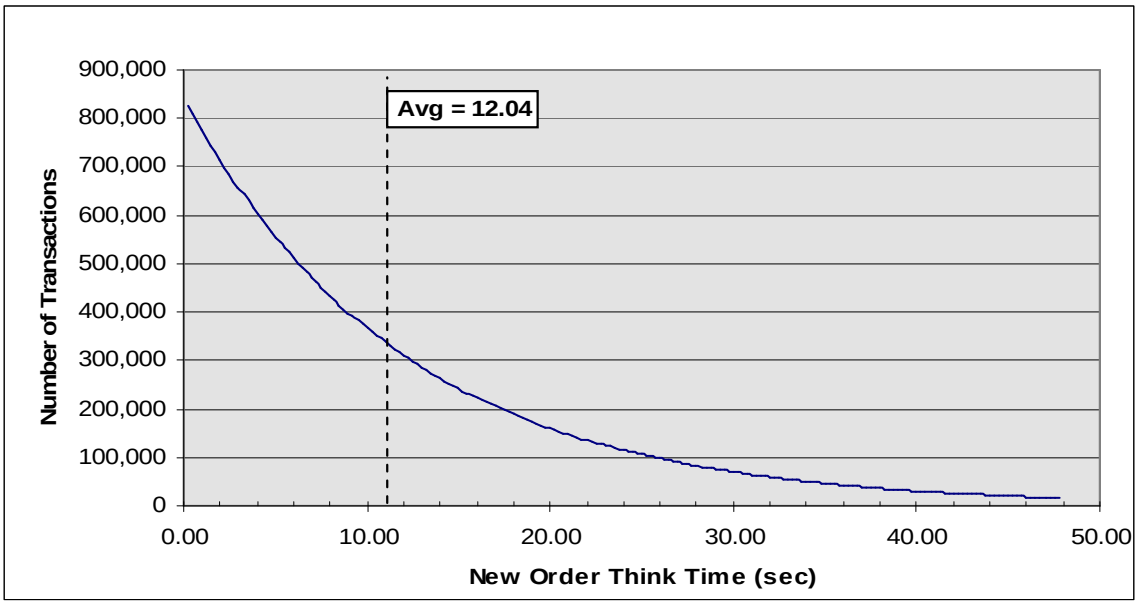


Figure 5-7: New-Order Think Time Distribution

5.6. Throughput versus Elapsed Time

A graph of throughput versus elapsed time must be reported for the New-Order transaction.

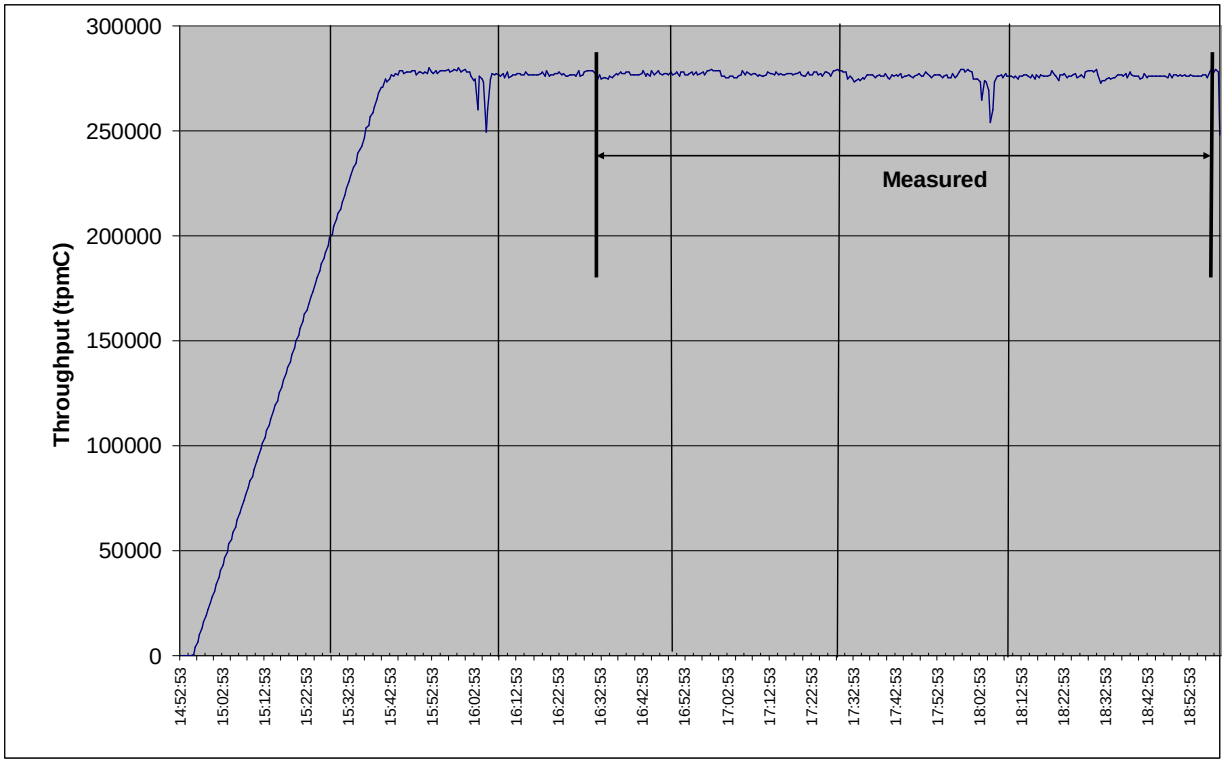


Figure 5-8: New-Order Throughput vs. Elapsed Time

5.7. Steady State Determination

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

All the emulated users were allowed to logon and do transactions. The user ramp-up phase is clearly visible on the graph above. Refer to the Numerical Quantities Summary pages for the rampup time. Figure 5-8 New-Order throughput versus Elapsed Time graph shows that the system maintained a steady state during the measurement interval

5.8. 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.

A 2 hour 28 minute measurement interval was used to guarantee that all work normally performed during an 8-hour sustained test is included in the reported throughput.

5.8.1. Transaction Flow

Each of the 4 (non-delivery) transactions is serviced by 2 individual programs, Internet Information System 6.0 (IIS) and a Microsoft COM+ 1.0 Queued Component Server, used as the transaction manager (COM+). Both programs are running on the client system:

- The initial HTML 1.0 request is serviced by an ISAPI custom-written handler running on Internet Information System 6.0. IIS is responsible for handling all HTML requests. The web server communicates to the COM+ server through a Microsoft COM+ API interface.

- COM+ communicates with the server system over Ethernet and handles all database operations**,** using Sybase embedded SQL calls.
  When the COM+ server boots up, it creates a configurable amount of connections to the server (listed in application settings).
  COM+ routes the transaction and balances the load according to the options defined in the Component Services GUI for the COM+ server application and settings in the Windows 2003 registry. The configuration file and registry variables are listed in Appendix B.2.
  At the beginning, each TPC-C user sends a pair of HTML 1.0 requests submitting its unique warehouse and district to the IIS ISAPI handler. Upon successful validation of user's login, IIS the displays an HTML form which encapsulates the TPC-C transaction menu.
  The transaction flow is described below:
  - The TPC-C user requests the transaction type's HTML form and proceeds to generate (fill in) a GET request with the required files for the transaction.
  - IIS accepts the filled in GET request , parses, and validates all values entered by the user.
  - It then proceeds to transmit those values to the COM+ server through an transaction type specific COM+ api interface.
  - The COM+ Pool Manager receives the request and first decides if there is a connection object in the pool available to service it.
    If so, the connection is used to send the transaction request to the Server.
    If no connection is available, the request will enter a COM+ internal queue and will be serviced by the next available connection.
  - Once the connection is available to be used, a COM+ pool thread receives the transaction and calls a TPC-C back end Sybase client api to execute all database operations related to the transaction type. (All the transaction information entered on the HTML form is available in a data structure provided by the ISAPI caller).
  - The transaction is committed and the Sybase back end client returns control back to the COM pool thread.
  - COM pool thread returns control to the ISAPI caller.
    (All transaction results are inside the data structure that the ISAPI caller provided to the COM+ api in the parameter list).
  - ISAPI caller returns control to the "screen application" by doing a PUT request.

5.8.2. Database Transaction

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

All database operations are performed by the TPC-C back-end programs. The process is described below:
Using ctlib calls, the TPC-C back-end program interacts with Sybase ASE Server to perform SQL data manipulations such as update, select, delete and insert, as required by the transaction. After all database operations are performed for a transaction, the transaction is committed.
Sybase ASE Server proceeds to update the database as follows:
When Sybase ASE Server changes a database table with an update, insert, or delete operation, the change is initially made in memory, not on disk. When there is not enough space in the memory buffer to read in or write additional data pages, Sybase ASE Server will make space by flushing some modified pages to disk. Modified pages are also written to disk as part of the checkpoint to ensure that no updates remain unflushed for longer than the allowed time. Before a change is made to the database, it is first recorded in the transaction log. This ensures that the database can be recovered completely in the event of a failure. Using the transaction log, transactions that started but did not complete prior to a failure can be undone, and transactions recorded as complete in the transaction log but not yet written to disk can be redone.

5.8.3. Checkpoints

A checkpoint is the process of writing all modified data pages to disk. The TPC-C benchmark was setup to automatically checkpoint every 29 minutes and 50 seconds. Three checkpoints occurs during the rampup period, and four other occurring during the measurement interval. The incremental checkpoint duration during the measurement is 29 minutes and 50 second.

Checkpoint	Start Time	End Time	Duration
	14:55:23	Run Begins	
#0	15:01:09	15:02:28	1:19
#1	15:30:59	15:54:04	23:05
#2	16:00:50	16:28:11	27:21
#3	16:30:40	16:57:05	26:25
	16:31:46	Measurement Interval Begins	
#4	17:00:31	17:27:12	26:41
#5	17:30:21	17:57:00	26:39
#6	18:00:11	18:27:02	26:51
#7	18:30:01	18:56:36	26:35
	18:59:46	Measurement Interval Ends	

5.9. Measurement Interval

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

A 2 hour 28 minute measurement interval was used. No connections were lost during the run.

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

6.1. RTE Availability

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

IBM used an internally developed RTE for these tests. 22,000 warehouses were configured. A rampup time of one hour thirty eight minutes was specified, along with a run time of two hours and 28 minutes.

6.2. Functionality and Performance of Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system.

No components were emulated.

6.3. Network Bandwidth

The bandwidth of the network(s) used in the tested/priced configuration must be disclosed.

The network between the clients and the database server was configured as 1000 MegaBits per second Full Duplex.

6.4. Operator Intervention

If the configuration requires operator intervention, the mechanism and the frequency of this intervention must be disclosed.

No operator intervention is required to sustain the reported throughput during the eight-hour period.

7 Clause 7: Pricing Related Items

7.1. Hardware and Programs Used

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

The detailed list of all hardware and programs for the priced configuration is listed in the pricing sheets for each system reported. The prices for all products and features that are provided by IBM are available the same day as product or feature availability.

7.2. Three Year Cost of System Configuration

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 pricing details for this disclosure is contained in the executive summary pages. All 3rd party quotations are included at the end of this report in Appendix D. All prices are based on IBM US list prices.

Discounts are based on US list prices and for similar quantities and configurations. A discount of 23.4% has been applied to specified IBM hardware, and services based on the total value and quantities of the components of the configuration.

7.3. Availability Dates

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

All components of the SUT will be available on December 16, 2008. Sybase ASE 15.0.3 is not immediately orderable. It will be available on December 16, 2008.

7.4. Statement of tpmC and Price/Performance

A statement of the measured tpmC, as well as the respective calculations for 3-year pricing, price/performance (price/tpmC), and the availability date must be disclosed.

System	tpmC	3-year System Cost	$/tpmC	Availability Date
IBM Power 550	276,383	$702,996 USD	$2.55 USD	December 16, 2008

Please refer to the price list on the Executive Summary page for details.

8 Clause 9: Audit Related Items

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

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Raymond J. Venditti
IBM Linux Performance
11501 Burnet Road
Austin, TX 78758

August, 2009

I verified the TPC Benchmark™ C performance of the following Client Server configuration. This result was initially published on June16, 2008. It is now revised to account for changes in the software and hardware pricing.

Platform: IBM Power 550 Express c/s
Operating system: Red Hat Enterprise Linux 5.1
Database Manager: Sybase ASE 15.0.3
Transaction Manager: Microsoft COM+

The results were:

CPU's Speed	Memory	Disks	New Order 90% Response Time	tpmC
Server: IBM Power 550 Express c/s				
2 x POWER6 (4.2GHz)	128 GB	490 x 73.4 GB 15K rpm SAS	0.70 Seconds	**276,383.3**
8 Client: IBM System x3250 (each with)				
1 x Intel Xeon X3210 QC (2.13 GHz)	1 GB (8 MB L2 cache)	1 x 73.4 GB SAS	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for the benchmark.

The following verification items were given special attention:

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

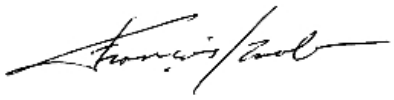
125 West Monroe Street • Colorado Springs, CO 80907 • 719-473-7555 • www.sizing.com

- The ACID properties were met
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured
- At least 90% of all delivery transactions met the 80 Second completion time limit
- All 90% response times were under the specified maximums
- The measurement interval was representative of steady state conditions
- The reported measurement interval was 148 minutes
- Four checkpoints were taken during the measurement interval
- The 60 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

The tested configuration included (8) priced clients model x3250 43658BU, each configured with 1 Quad-Core Intel Xeon X3210 (2.13GHz/1066MHz FSB, 2x4MB L2). Having reached end-of-life, these clients were substituted in the new priced configuration by (8) x3350 419284U, each configured with 1 Quad-Core Intel Xeon Processor X3360 (2.83GHz/1333MHz- 2x6MB L2). Based on data analysis done for each type of client and the TPC requirements for such subnstitution, it is my opinion that this substitution has no significant effect on performance.

Respectfully Yours,



François Raab, President

125 West Monroe Street • Colorado Springs, CO 80907 • 719-473-7555 • www.sizing.com

10 Appendix A: Client Server Code

10.1. Client/Terminal Handler Code

htmlPhraser.cpp

```
//////////////////////////////////////////////////////////////////////
// htmlPhraser.cpp
//////////////////////////////////////////////////////////////////////
// Class implmentation of htmlPhraser.
// This class will take a query string and break it into a series
// of consituant parts
//////////////////////////////////////////////////////////////////////

#include "htmlPhraser.h"

//////////////////////////////////////////////////////////////////////
// htmlPhraser::htmlPhraser
//////////////////////////////////////////////////////////////////////
// Title   : Constructor
// Parameters : char * query string
// Return Value  : None
// Comments  :
//////////////////////////////////////////////////////////////////////

htmlPhraser::htmlPhraser(char *queryString)
{
   // initilize query values
   iCustomerIdFlag = iCarrierNumFlag = iStockThresholdFlag
= false;

   // this initilizes the query list to NULL's. This means that
   // characters being added are overwriting null characters
and
   // therfore the string will be null terminated implicitly.

   memset(iQueryValues,NULL,(MAX_FIELD_NUM *
MAX_FIELD_LEN));

   // controls
   char      queryChar      = NULL;

   int      queryIndex     = -1;
   int      valueIndex     = -1;

   // process each characeter of query string
   while(*queryString)
   {
      // check for special case characters
      if(queryChar)
      {
         // a percentage sign would indicate a token
         if(*queryString != '%')
         {
            // a plus sign repersents a space
            if(*queryString == '+')
            {
               queryChar = ' ';
               *queryString++;
            }
            else queryChar = *queryString++;
         }
         else queryChar = convertQueryToken(&queryString);
      }
      else queryChar = '&';

      // handle query reference (&)
      if(queryChar == '&')
      {
         // reset value index
         valueIndex = -1;

         // do we have a numeric query reference
         if(*queryString >= '0' && *queryString <= '9')
         {
            // numeric query id
            queryIndex =
               ((*queryString - '0') * 10) + (*(queryString + 1) - '0');

            // walk past the two command characters
            queryString += 2;

            // validate query value
            if(queryIndex > MAX_QUERY_ID)
               queryIndex = -1;
         }
         else queryIndex = -1;

         // finished processing for query reference
         continue;
      }

      // we have a query reference but need to wait until we
see '='
      // before accepting value
      if(valueIndex == -1)
      {
         // we are waiting for '='
         if(queryChar == '=')
         {
            valueIndex = 0;

            // set query string flags
            switch(queryIndex)
            {
            case C_ID:
               iCustomerIdFlag = true; break;
            case CARRIER_NUM:
               iCarrierNumFlag = true; break;
            case STK_THRESHOLD:
               iStockThresholdFlag = true; break;
            default: break;
            }
         }

         // finishes looging for '='
         continue;
      }

      // add each character to the query value
      if(queryIndex > -1 && valueIndex > -1)
      {
         // we are processing a query value
         if(valueIndex < MAX_FIELD_LEN)
         {
            // we have not exceeded max line len
            iQueryValues[queryIndex][valueIndex++] =
queryChar;
         }
         continue;
      }
   }

   return;
}

//////////////////////////////////////////////////////////////////////
// htmlPhraser::getCommandId
//////////////////////////////////////////////////////////////////////
// Title   : Returns the page command
// Parameters : None
// Return Value  : int - page command
// Comments  :
//////////////////////////////////////////////////////////////////////

int htmlPhraser::getCommandId()
{
```

```
   // return command numeric code
   switch(*iQueryValues[COMMAND_ID])
   {
   case NEW_ORDER_CODE:
      if(iCustomerIdFlag)
         return COMMAND_NEW_ORDER_RESULTS;
      else return COMMAND_NEW_ORDER;
   case PAYMENT_CODE:
      if(iCustomerIdFlag)
         return COMMAND_PAYMENT_RESULTS;
      else return COMMAND_PAYMENT;
   case ORDER_STATUS_CODE:
      if(iCustomerIdFlag)
         return COMMAND_ORDER_STATUS_RESULTS;
      else return COMMAND_ORDER_STATUS;
   case DELIVERY_CODE:
      if(iCarrierNumFlag)
         return COMMAND_DELIVERY_RESULTS;
      else return COMMAND_DELIVERY;
   case STOCK_CODE:
      if(iStockThresholdFlag)
         return COMMAND_STOCK_RESULTS;
      else return COMMAND_STOCK;
   case MENU_CODE:
      return COMMAND_LOGIN_RESULTS;
   case EXIT_CODE:
      return COMMAND_EXIT;
   default:
      return COMMAND_LOGIN;
   };


   // should not get here
   return COMMAND_LOGIN;
}

//////////////////////////////////////////////////////////////////////
// htmlPhraser::validate
//////////////////////////////////////////////////////////////////////
// Title   : validate url parameter list for all txn types
// Parameters : int - txn type
// Return Value  : int - error code
// Comments  :
//////////////////////////////////////////////////////////////////////


int validate(int txnType)
{
   return 0;
}

//////////////////////////////////////////////////////////////////////
```

```
// htmlPhraser::convertQueryToken
//////////////////////////////////////////////////////////////////////
// Title   : Returns the page command
// Parameters : None
// Return Value  : int - page command
// Comments  :
//////////////////////////////////////////////////////////////////////

char htmlPhraser::convertQueryToken(char **queryString)
{
   char queryChar = NULL;

   // skip over %
   (*queryString)++;

   // look at first character
   switch(**queryString)
   {
   case '2':
      {
         // what follows?
         (*queryString)++;

         switch(**queryString)
         {
         case '1':
            queryChar = '!';
            break;
         case '3':
            queryChar = '#';
            break;
         case '4':
            queryChar = '$';
            break;
         case '5':
            queryChar = '%';
            break;
         case '6':
            queryChar = '&';
            break;
         case '8':
            queryChar = '(';
            break;
         case '9':
            queryChar = ')';
            break;
         case 'B':
            queryChar = '+';
            break;
         case 'C':
            queryChar = ',';
            break;
         case 'F':
            queryChar = '/';
            break;
         case ' ':
            queryChar = ' ';
            break;
         }
      }

      break;
   case '3':
      {
         // what follows?
         (*queryString)++;

         switch(**queryString)
         {
         case 'A':
            queryChar = ':';
            break;
         case 'B':
            queryChar = ';';
            break;
         case 'D':
            queryChar = '=';
            break;
         case 'F':
            queryChar = '?';
            break;
         case ' ':
            queryChar = ' ';
            break;
         }
      }

      break;
   case '4':
      {
         // what follows?
         (*queryString)++;

         switch(**queryString)
         {
         case '0':
            queryChar = '@';
            break;
         case ' ':
            queryChar = ' ';
            break;
         }
      }
```

```
    break;
  case '5':
    {
      // what follows?
      (*queryString)++;

      switch(**queryString)
      {
      case 'B':
        queryChar = '[';
        break;
      case 'D':
        queryChar = ']';
        break;
      case 'E':
        queryChar = '^';
        break;
      case ' ':
        queryChar = ' ';
        break;
      }
    }

    break;
  case '7':
    {
      // what follows?
      (*queryString)++;

      switch(**queryString)
      {
      case 'B':
        queryChar = '{';
        break;
      case 'C':
        queryChar = '|';
        break;
      case 'D':
        queryChar = '}';
        break;
      case 'E':
        queryChar = '~';
        break;
      case ' ':
        queryChar = ' ';
        break;
      }
    }

    break;
  case '+':
    queryChar = '+';
    break;
  }

  // advance pointer and return
  (*queryString)++; return queryChar;
}

//////////////////////////////////////////////////////////////////
```

htmlPhraser.h

```
//////////////////////////////////////////////////////////////////
// htmlPharaser.h
//////////////////////////////////////////////////////////////////
// Class to decode a html query string
//////////////////////////////////////////////////////////////////

#pragma once

#include <memory.h>

//////////////////////////////////////////////////////////////////
// Definitions
//////////////////////////////////////////////////////////////////

#define NULL            0

#define COMMAND_ID          0
#define TERM_ID         1
#define W_ID            2
#define D_ID            3
#define C_ID            4
#define C_NAME          5

#define C_W_ID          6
#define C_D_ID          7
#define AMT_PAID        8

#define STK_THRESHOLD       9
#define CARRIER_NUM         10

#define ITEM_LIST_START     11
#define ITEM_LIST_FINISH    55

#define MAX_QUERY_ID        55
#define MAX_FIELD_LEN       256
#define MAX_FIELD_NUM       56

//////////////////////////////////////////////////////////////////
// Command Codes
//////////////////////////////////////////////////////////////////

#define NEW_ORDER_CODE          'n'
#define PAYMENT_CODE            'p'
#define ORDER_STATUS_CODE       'o'
#define DELIVERY_CODE           'd'
#define STOCK_CODE              's'
#define EXIT_CODE               'e'
#define MENU_CODE               'm'

#define COMMAND_LOGIN           0
#define COMMAND_NEW_ORDER       1
#define COMMAND_PAYMENT         2
#define COMMAND_ORDER_STATUS    3
#define COMMAND_DELIVERY        4
#define COMMAND_STOCK           5
#define COMMAND_EXIT            6

#define COMMAND_LOGIN_RESULTS       7
#define COMMAND_NEW_ORDER_RESULTS   8
#define COMMAND_PAYMENT_RESULTS     9
#define COMMAND_ORDER_STATUS_RESULTS    10
#define COMMAND_DELIVERY_RESULTS    11
#define COMMAND_STOCK_RESULTS       12

//////////////////////////////////////////////////////////////////
// Class htmlPhraser
//////////////////////////////////////////////////////////////////

class htmlPhraser
{
  // Constructors / Destructor
  public:
    htmlPhraser(char *queryString);
    ~htmlPhraser()          {return;}

   // geters
   public:
     int   getCommandId();
     int validate(int txnType);

     char *get_TERM_ID()    {return
iQueryValues[TERM_ID];}
     char *get_W_ID()       {return iQueryValues[W_ID];}
     char *get_D_ID()       {return iQueryValues[D_ID];}
     char *get_C_ID()       {return iQueryValues[C_ID];}
     char *get_C_NAME()     {return
iQueryValues[C_NAME];}
     char *get_C_W_ID()     {return
iQueryValues[C_W_ID];}
     char *get_C_D_ID()     {return iQueryValues[C_D_ID];}
     char *get_AMT_PAID()   {return
iQueryValues[AMT_PAID];}
```

```
    char *get_STK_THRESHOLD()    {return
iQueryValues[STK_THRESHOLD];}
    char *get_CARRIER_NUM()     {return
iQueryValues[CARRIER_NUM];}

    char *get_ITEM_SUPP_W(int item)  {return
iQueryValues[(ITEM_LIST_START + 0) + (item * 3)];}
    char *get_ITEM_ITEM_NUM(int item) {return
iQueryValues[(ITEM_LIST_START + 1) + (item * 3)];}
    char *get_ITEM_QTY(int item)     {return
iQueryValues[(ITEM_LIST_START + 2) + (item * 3)];}

  // Class Functions
  private:
    char convertQueryToken(char **queryString);

  // Class Attributes
  private:
    int   iCustomerIdFlag;
    int   iCarrierNumFlag;
    int   iStockThresholdFlag;

    char
  iQueryValues[MAX_FIELD_NUM][MAX_FIELD_LEN];
};

////////////////////////////////////////////////////////////////////////
```

resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by tpccIsapi.rc
//
#define IDS_PROJNAME 100



// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE      201
#define _APS_NEXT_COMMAND_VALUE       32768
#define _APS_NEXT_CONTROL_VALUE       201
#define _APS_NEXT_SYMED_VALUE         101
#endif
#endif
```

StdAfx.cpp

```
// stdafx.cpp : source file that includes just the standard
includes
// tpccIsapi.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in
STDAFX.H
// and not in this file
```

StdAfx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once

#define WIN32_LEAN_AND_MEAN    // Exclude rarely-used
stuff from Windows headers

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS //
some CString constructors will be explicit

// turns off ATL's hiding of some common and often safely
ignored warning messages
#define _ATL_ALL_WARNINGS

// critical error descriptions will only be shown to the user
// in debug builds. they will always be logged to the event log
#ifndef _DEBUG
#define ATL_CRITICAL_ISAPI_ERROR_LOGONLY
#endif


#ifndef _WIN32_WINNT
#define _WIN32_WINNT 0x0403
#endif

// TODO: this disables support for registering COM objects
// exported by this project since the project contains no
// COM objects or typelib. If you wish to export COM objects
// from this project, add a typelib and remove this line
#define _ATL_NO_COM_SUPPORT

#include "resource.h"
#include <atlsrvres.h>
#include <atlisapi.h>
#include <atlstencil.h>

// TODO: reference additional headers your program requires
here
```

time.cpp

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/timeb.h>
#include <time.h>

char *get_time_prefix(char *buffer)
{
   time_t cur_timet;
   char time_str[30];
   int len;

   cur_timet = time(&cur_timet);
   strftime(time_str, 29, "%X", localtime(&cur_timet));

   len = sprintf(buffer, "%s - ",
            time_str);
   if (len >= 30) {
     sprintf(buffer, "too small: %d\n",
           30, len);
   }
   return(buffer);
}
```

tpcc.h

```
// Common defines and structures use internally by client
code
// Not to be confused with structures actually passed in
transaxtions
//

// standard includes

#ifndef _COMMON_TPCC
#define _COMMON_TPCC

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/timeb.h>
#include <time.h>



#ifdef DB2
  #include <db2tpcc.h>
```

```
#endif

#ifdef ORACLE
  #include <ora_tpcc.h>
#endif

#ifdef SYBASE
  #include <sybtpcc.h>
#endif


#include <iostream>
#include <fstream>
#include <process.h>
#include <ios>

//////////////////////////////////////////////////////////////
// Defines
//////////////////////////////////////////////////////////////

#define   OK              0
#define   INVALID_STATUS      -1
#define   ERR             -1
#define   INVALID_COM_STATUS    -2


#define TXN_MAX_COMMANDS      55
#define MAX_TRANSACTIONS      14
#define MAX_CMD_LENGTH        100
#define INPUT_ITEMS           3
#define MAX_INT_BUFFER        15
#define NORD_ITEMS            15
#define ITEM_START            11
#define ITEM_END              55
#define MAX_ITEMS             15

#define MAX_STRING_LEN        256
#define MAX_HTML_PAGE_LEN     4096
#define MAX_HTML_HEADER_LEN       512

#define DELIVERY_THREADS_NUM    100

#define DISTRICTS_PER_WAREHOUSE   10
//////////////////////////////////////////////////////////////
// Transaction Codes
//////////////////////////////////////////////////////////////

#define   TXN_LOGIN           0
#define   TXN_NEW_ORDER         1
#define   TXN_PAYMENT           2
#define   TXN_ORDER_STATUS      3
#define   TXN_DELIVERY          4
#define   TXN_STOCK             5
#define   TXN_EXIT              6
#define   TXN_LOGIN_RESULTS       7
#define   TXN_NEW_ORDER_RESULTS   8
#define   TXN_PAYMENT_RESULTS     9
#define   TXN_ORDER_STATUS_RESULTS 10
#define   TXN_DELIVERY_RESULTS    11
#define   TXN_STOCK_RESULTS       12

#define   CMD_NORD            "nord"
#define   CMD_PYMT            "pymt"
#define   CMD_ORDS            "ords"
#define   CMD_DLVY            "dlvy"
#define   CMD_STOK            "stok"
#define   CMD_EXIT            "exit"
#define   CMD_MENU            "menu"

#define   APP_NAME            "tpcc.html"
#define   HEADER              "Content-
Type:text/html\r\nContent-Length: %d\r\nConnection: Keep-
Alive\r\n\r\n"

//////////////////////////////////////////////////////////////
// URL Commands
//////////////////////////////////////////////////////////////

#define   CMD_TXN_ID          "00"
#define   CMD_TERM_ID           "01"
#define   CMD_W_ID        "02"
#define   CMD_D_ID        "03"
#define   CMD_C_ID        "04"
#define   CMD_C_NAME          "05"
#define   CMD_C_W_ID          "06"
#define   CMD_C_D_ID          "07"
#define   CMD_AMT_PAID        "08"
#define   CMD_STK_THRESHOLD   "09"
#define   CMD_CARRIER_NUM       "10"

#define   ITEM01_SUPP_W       "11"
#define   ITEM01_ITEM_NUM       "12"
#define   ITEM01_OTY        "13"


#define CHAR_FILL           ' '
#define NUMERIC_FILL        ' '
#define NEGITIVE_SYMBOL           '-'
#define MONEY_SYMBOL          '$'
#define DECIMAL_SYMBOL            '.'
#define ZERO_SYMBOL           '0'
#define ZIP_DELIMITER         '-'
#define PHONE_DELIMITER           '-'
#define DATE_DELIMITER          '-'
#define TIME_DELIMITER          ':'

#define DEFAULT_MONEY64_LEN     15
#define DEFAULT_MONEY32_LEN     9
#define DEFAULT_MONEY16_LEN     9

#define DEFAULT_NUMERIC64_LEN   15
#define DEFAULT_NUMERIC32_LEN   9
#define DEFAULT_NUMERIC16_LEN   9

#define DEFAULT_DECIMAL64_LEN   5
#define DEFAULT_DECIMAL32_LEN   5
#define DEFAULT_DECIMAL16_LEN   5

#define DEFAULT_DATETIME_LEN    19
#define DEFAULT_DATE_LEN      11
#define DEFAULT_TIME_LEN      8

#define DEFAULT_STRING_LEN      25
#define DEFAULT_ZIP_LEN         17
#define DEFAULT_PHONE_LEN       18

//////////////////////////////////////////////////////////////////////
// String Field Lengths
//////////////////////////////////////////////////////////////////////

#define NAME_LEN    24
#define LAST_NAME_LEN   16
#define FIRST_NAME_LEN  16
#define INITIALS_LEN    2

#define CREDIT_LEN    2

#define STREET_LEN      20
#define CITY_LEN      20
#define STATE_LEN     2
#define ZIP_LEN       9

#define PHONE_LEN     16
#define DATA_LEN      200

#define ITEM_LIST     15
#define ORDER_LIST    10

//////////////////////////////////////////////////////////////////////
// Type definitions
//////////////////////////////////////////////////////////////////////

typedef __int8      INT8b;
typedef __int16       INT16b;
typedef __int32       INT32b;
typedef __int64       INT64b;
```

```
typedef unsigned __int8    UINT8b;
typedef unsigned __int16   UINT16b;
typedef unsigned __int32   UINT32b;
typedef unsigned __int64   UINT64b;

typedef INT16b          sqlint16;
typedef INT32b          sqlint32;
typedef INT64b          sqlint64;

typedef INT16b          int16_t;
typedef INT32b          int32_t;
typedef INT64b          int64_t;

typedef char            BYTE8b;
typedef double          DOUBLE;
typedef unsigned long   NATURAL;

//////////////////////////////////////////////////////////////////
// Date and time values
//////////////////////////////////////////////////////////////////

#define SECONDS_IN_DAY      86400
#define SECONDS_IN_HOUR     3600
#define SECONDS_IN_MINUTE   60
#define GMT_OFFSET          5

#define DAYS_IN_YEAR    365
#define YEARS_IN_LEAP   4
#define START_YEAR      1970
#define MONTHS_IN_YEAR  12

//////////////////////////////////////////////////////////////////
// Error codes
//////////////////////////////////////////////////////////////////
#define ERR_INVALID_TXN_TYPE        -1

#define ERR_MISSING_W_ID            -2
#define ERR_NON_NUMERIC_W_ID        -3
#define ERR_MISSING_D_ID            -4
#define ERR_NON_NUMERIC_D_ID        -5
#define ERR_MISSING_C_ID            -6
#define ERR_NON_NUMERIC_C_ID        -7

#define ERR_MISSING_SUPP_W          -8
#define ERR_NON_NUMERIC_SUPP_W      -9
#define ERR_MISSING_ITEM_NUM        -10
#define ERR_NON_NUMERIC_ITEM_NUM    -11
#define ERR_MISSING_ITEM_OTY        -12
#define ERR_NON_NUMERIC_ITEM_QTY    -13

#define ERR_MISSING_CLAST_NAME      -14
#define ERR_NON_NUMERIC_CUST_W_ID   -15
#define ERR_NON_NUMERIC_CUST_D_ID   -16
#define ERR_MISSING_AMOUNT_PAID     -17
#define ERR_NON_NUMERIC_AMOUNT_PAID -18

#define ERR_INVALID_D_ID        "ERROR : Invalid District
ID. Try Again."
#define ERR_INVALID_W_ID        "ERROR : Invalid
Warehouse ID. Try Again."
#define ERR_INVALID_C_ID        "ERROR : Invalid
Customer ID. Try Again."
#define ERR_INVALID_SUPPLY_W_ID "ERROR : Invalid
Item Supply Warehouse. Try Again."
#define ERR_INVALID_ITEM_NUM    "ERROR : Invalid Item
Number. Try Again."
#define ERR_INVALID_ITEM_OTY    "ERROR : Invalid Item
Qty. Try Again."
#define ERR_MISSING_C_ID_OR_CLAST  "ERROR : Must
Enter Customer Id or Customer Last Name. Try Again."
#define ERR_INVALID_PAYMENT_AMOUNT "ERROR :
Invalid Payment Amount. Try Again."
#define ERR_INVALID_CARRIER     "ERROR : Invalid
Carrier Number. Try Again."
#define ERR_INVALID_THRESHOLD   "ERROR : Invalid
Threshold. Try Again."
#define ERR_INVALID_C_D_ID      "ERROR : Invalid
Customer District Id. Try Again."
#define ERR_INVALID_C_W_ID      "ERROR : Invalid
Customer Warehouse Id. Try Again."
#define ERR_TERMINAL_FULL       "ERROR : Terminal can
not support user.  Terminal full."
#define ERR_C_ID_OR_CLAST_ONLY  "ERROR : Either
customer id or customer last name can be specified."

#define ERR_UNABLE_TO_OPEN_REG      -50
#define ERR_DLVY_THREAD_FAILED      -51
#define ERR_DLVY_SEMAPHORE_INIT_FAILED -52
#define ERR_DLVY_EVENT_INIT_FAILED  -53
#define ERR_DLVY_QUEUE_EATING_TAIL  -54
#define ERR_DLVY_QUEUE_CALLOC_FAIL  -55

#define ERR_INVALID_USERNAME        -70
#define ERR_INVALID_PASSWORD        -71
#define ERR_INVALID_DB_NAME         -72
#define ERR_INVALID_REGISTERY_KEY   -73
#define ERR_DB2_DLL_NOT_LOADED      -74
#define ERR_ORACLE_DLL_NOT_LOADED   -75
#define ERR_CONNECT_ADDRESS_NOT_FOUND-76
#define ERR_NORD_ADDRESS_NOT_FOUND  -77
#define ERR_PYMT_ADDRESS_NOT_FOUND  -78
#define ERR_ORDS_ADDRESS_NOT_FOUND  -79
#define ERR_DLVY_ADDRESS_NOT_FOUND  -80
#define ERR_STOK_ADDRESS_NOT_FOUND  -81
#define ERR_NULL_DLL_NOT_LOADED     -82
#define ERR_UNKNOWN_DB              -83
#define ERR_DISCONNECT_ADDRESS_NOT_FOUND -84
#define ERR_ORA_DLL_NOT_LOADED      -85
#define ERR_SYB_DLL_NOT_LOADED      -86
#define ERR_DBINIT_ADDRESS_NOT_FOUND -87
#define ERR_INVALID_APPNAME         -88

#define ERR_SAVING_CONTEXT          -90
#define ERR_DETACHING_CONTEXT       -91
#define ERR_ATTACHING_CONTEXT       -92
#define ERR_HANDLE_IN_USE           -93

#define ERR_CONNECT_TO_TM_FAILED    -99
#define ERR_DLVY_LOG_OPEN_FAILED    -100
#define ERR_DLVY_QUEUE_FULL         -101

//////////////////////////////////////////////////////////////////
// Registery Definitions
//////////////////////////////////////////////////////////////////
#define   REGISTERY_SUB_KEY   "SOFTWARE\\TPCC"

#define   DELIVERY_THREADS    "dlvyThreads"
#define   DELIVERY_QUEUE_LEN  "dlvyQueueLen"
#define   DELIVERY_LOG_PATH   "dlvyLogPath"
#define   ERROR_LOG_FILE      "errorLogFile"
#define   HTML_TRACE_LOG_FILE "htmlTraceLogFile"
#define   DB_NAME             "dbName"
#define   NULL_DB             "nullDB"
#define   COM_NULL_DB         "comnullDB"
#define   CLIENT_NULL_DB      "clientNullDB"

#define   NUM_USERS           "numUsers"
#define   DB_TYPE             "dbType"

#define   TXN_MONITOR         "txn_server"
#define   COMM_POOL           "comm_pool"
#define   HTML_TRACE          "htmlTrace"
#define   ISAPI_TRACE         "isapi_trace"

#define   DEFAULT_DLVY_THREADS    1
#define   DEFAULT_DLVY_QUEUE_LEN  10
#define   DEFAULT_DLVY_LOG_PATH
  "c:\\inetpub\\wwwroot\\tpcc\\dlvy"
#define   DEFAULT_ERROR_LOG_FILE
  "c:\\inetpub\\wwwroot\\tpcc\\errorLog.txt"
#define   DEFAULT_HTML_TRACE_LOG_FILE
  "c:\\inetpub\\wwwroot\\tpcc\\htmlTrace.txt"
#define   DEFAULT_NUM_USERS       10000

#define   DEFAULT_DB_NAME         "tpcc"
```

```
//////////////////////////////////////////////////////////////////
// Structure defines
//////////////////////////////////////////////////////////////////

struct nord_wrapper {
     in_neword_type  in_nord;
     out_neword_type   out_nord;
};

struct paym_wrapper {
   struct in_payment_struct in_paym;
   struct out_payment_struct out_paym;
};

struct ords_wrapper {
   struct in_ordstat_struct in_ords;
   struct out_ordstat_struct out_ords;
};

struct dlvy_wrapper {
   struct in_delivery_struct in_dlvy;
   struct out_delivery_struct out_dlvy;
};

struct stok_wrapper {
   struct in_stocklev_struct in_stok;
   struct out_stocklev_struct out_stok;
};

typedef struct
{
   int  year;
   int  month;
   int  day;

   int  hour;
   int  minute;
   int  second;
} datetime;

struct NEWORDERDATA
{
   struct   in_items_struct {
      int  s_OL_I_ID;
      int  s_OL_SUPPLY_W_ID;
      short  s_OL_QUANTITY;
   } in_item[15];

   long long  in_s_O_ENTRY_D_time;  /* init by SUT */
   int      in_s_C_ID;
   int      in_s_W_ID;
   short  in_s_D_ID;
   short  in_s_O_OL_CNT;    /* init by SUT */
   short  in_s_all_local;
   short  in_duplicate_items;

   struct   out_items_struct {
      float   s_I_PRICE;
      float  s_OL_AMOUNT;
      short  s_S_QUANTITY;
      char    s_I_NAME[25];
      char    s_brand_generic;
   } out_item[15];

   long long  out_s_O_ENTRY_D_time;
   double  out_s_W_TAX;
   double  out_s_D_TAX;
   double  out_s_C_DISCOUNT;
   double  out_s_total_amount;
   int      out_s_O_ID;
   short  out_s_O_OL_CNT;
   short  out_s_transtatus;
   short  out_deadlocks;
   char    out_s_C_LAST[17];
   char    out_s_C_CREDIT[3];

};

struct PAYMENTDATA
{
   long long  in_s_H_DATE_time;
   double  in_s_H_AMOUNT;
   int      in_s_W_ID;
   int      in_s_C_W_ID;
   int      in_s_C_ID;
   short  in_s_C_D_ID;
   short  in_s_D_ID;
   char   in_s_C_LAST[17];

   long long  out_s_H_DATE_time;
   long long  out_s_C_SINCE_time;
   double  out_s_C_CREDIT_LIM;
   double  out_s_C_BALANCE;
   double  out_s_C_DISCOUNT;
   int      out_s_C_ID;
   short  out_s_transtatus;
   short  out_deadlocks;
   char  out_s_W_STREET_1[21];
   char  out_s_W_STREET_2[21];
   char  out_s_W_CITY[21];
   char  out_s_W_STATE[3];
   char  out_s_W_ZIP[10];
   char  out_s_D_STREET_1[21];
   char  out_s_D_STREET_2[21];
   char  out_s_D_CITY[21];
   char  out_s_D_STATE[3];
   char  out_s_D_ZIP[10];
   char  out_s_C_FIRST[17];
   char  out_s_C_MIDDLE[3];
   char  out_s_C_LAST[17];
   char  out_s_C_STREET_1[21];
   char  out_s_C_STREET_2[21];
   char  out_s_C_CITY[21];
   char  out_s_C_STATE[3];
   char  out_s_C_ZIP[10];
   char  out_s_C_PHONE[17];
   char  out_s_C_CREDIT[3];
   char  out_s_C_DATA[201];
};

struct ORDERSTATUSDATA
{
   int   in_s_C_ID;
   int   in_s_W_ID;
   short in_s_D_ID;
   char  in_s_C_LAST[17];

   double out_s_C_BALANCE;
   long long  out_s_O_ENTRY_D_time;
   int    out_s_C_ID;
   int    out_s_O_ID;
   short  out_s_O_CARRIER_ID;
   short  out_s_ol_cnt;
   struct out_oitems_struct {
      long long  s_OL_DELIVERY_D_time;
      float   s_OL_AMOUNT;
      int   s_OL_I_ID;
      int   s_OL_SUPPLY_W_ID;
      short s_OL_QUANTITY;
   } out_item[15];
   short  out_s_transtatus;
   short  out_deadlocks;
   char   out_s_C_FIRST[17];
   char   out_s_C_MIDDLE[3];
   char   out_s_C_LAST[17];

};

struct DELIVERYDATA
{
   long long  in_s_O_DELIVERY_D_time;
   int   in_s_W_ID;
   short in_s_O_CARRIER_ID;
   int   out_s_O_ID[10];
   short out_s_transtatus;
```

```
   short outdeadlocks;
};

struct STOCKLEVELDATA
{
   int   in_s_threshold;
   int   in_s_W_ID;
   short in_s_D_ID;

   int   out_s_low_stock;
   short out_s_transtatus;
   short out_deadlocks;
};

struct DLVYQUEUEDATA
{
   int      warehouse;
   short  in_s_0_CARRIER_ID;

   struct _timeb enqueueTime;
};


//              MISCELLANEOUS HELPER FUNCTIONS
inline void appendText(char **string,char *text);
inline void appendText(char **string,char *text,int length,int
justify);
inline void appendChar(char **string,char byte);
inline void DEBUGMSG(FILE * debugFile, char * message);
inline void appendSpaces(char **string,int spaces);

inline void calcOutDateTime(const INT64b value,datetime
*timestamp);
inline int copyOutPhone(char *buffer,char *value,int len);
inline bool copyInMoney64(const char * value,INT64b
*number);
inline bool copyInMoney32(const char * value,int *number);
inline int copyInMoney(const char *value);
inline void copyOutMoney64(char *buffer,INT64b
value,unsigned int len);
inline int copyOutDateTime(char*buffer,INT64b value);
inline int copyOutDate(char *buffer,INT64b value);
inline int copyOutTime(char *buffer,INT64b value);
inline int copyOutDecimal64(char *buffer,INT64b
value,unsigned int len);

inline UINT16b changeOrder16(UINT16b value);
inline UINT32b changeOrder32(UINT32b value);
inline UINT64b changeOrder64(UINT64b value);

inline INT16b changeOrder16(INT16b value);
inline INT32b changeOrder32(INT32b value);
inline INT64b changeOrder64(INT64b value);

//
// Name         : appendText
// Description  :
//              Append text to string
// Parameters   :
//              char ** - string point to append to
//              char * - text to append
// Returns      :
//              None
// Comments     :
//

inline void appendText(char **string,char *text)
{
   while(*text)
   {
      *(*string)++ = *text++;
   }

   **string='\0';
   return;
}

//
// Name         : appendText
// Description  :
//              Append text to string
// Parameters   :
//              char ** - string point to append to
//              char * - text to append
//            int - total field length including blank spaces
//            int - justify flag
// Returns      :
//              None
// Comments     :
//            right justify
//            left justify


inline void appendText(char **string,char *text,int length,int
justify)
{
   int byteCount = 0;

   if(justify)
   {
      while(*text)
      {
         *(*string)++ = *text++;
         byteCount++;
      }

      //append blank spaces if text is less than length at end
      for(byteCount;byteCount < length;byteCount++)
         *(*string)++ = ' ';
   }
   else
   {
      long long textLen = strlen(text);
      for(textLen;textLen < length;textLen++)
         *(*string)++ = ' ';

      while(*text)
         *(*string)++ = *text++;

   }
   **string='\0';
}


// Name              : appendChar
// Description  :
//                              Append text to string
// Parameters   :
//                              char ** - string point to append to
//                              char * - text to append
// Returns           :
//                              None
// Comments          :
//

inline void appendChar(char **string,char byte)
{
   *(*string)++ = byte;
   **string='\0';

   return;
}


//
// Name       : appendSpaces
// Description  :
//            appends buffer spaces to result page
// Parameters :
//              **htmlPage
//
// Returns    :
//              amount of characters the function appened
//              to the html page
// Comments     :
//
```

```
inline void appendSpaces(char **string,int spaces)
{
   for(int index=0;index<spaces;index++)
   {
      *(*string)++ = ' ';
   }

   **string='\0';
}

//
// Name       : appendCustData
// Description :
//            appends cust data buffer to result page
// Parameters :
//               **htmlPage
//
// Returns    :
//
//               Adds a newline character every 50 characters
displayed.
// Comments     :
//

inline void appendCustData(char **string,char *text)
{
   short byteCount = 0;
   while(*text)
   {
      *(*string)++ = *text++;
      byteCount++;
      if((byteCount % 50) == 0)
      {
         *(*string)++ = '\n';
         *(*string)++ = ' ';*(*string)++ = ' ';*(*string)++ = ' ';
   *(*string)++ = ' ';
         *(*string)++ = ' ';*(*string)++ = ' ';*(*string)++ = ' ';
   *(*string)++ = ' ';
         *(*string)++ = ' ';*(*string)++ = ' ';*(*string)++ = ' ';
      }
   }
   **string='\0';
}



//
// calcOutDateTime
//
// Title      : Calculate date & time data out of class array
// Parameters : INT64b - date & time expressed in seconds
//            datetime * - timestamp
// Return Value  : None
// Comments      :
//

inline void calcOutDateTime(const INT64b value,datetime
*timestamp)
{
   // fixed days in each month (FEB 29 is special case)
   static int daysInMonth[12] =
{31,28,31,30,31,30,31,31,30,31,30,31};

   // mask out EPOC seconds
   int dateValue  = ((int) (value & 0xffffffff)) +
            (SECONDS_IN_DAY - (GMT_OFFSET *
SECONDS_IN_HOUR));

   int offset   = (int) (value >> 32);

   // break out the seconds
   int hms     = dateValue % SECONDS_IN_DAY;
   int days= dateValue / SECONDS_IN_DAY;

   int years   = (days - 1) / DAYS_IN_YEAR;
   int leaps   = years / YEARS_IN_LEAP;

   int daysUsed  = (years * DAYS_IN_YEAR) + leaps;

   // adjust the number of days to account for calculated years
   days = days - daysUsed;

   // set the starting year, month, and day
   timestamp->day    = 1;
   timestamp->month  = 1;
   timestamp->year      = START_YEAR + years;

   // is the current year a leap year
   int leap = !(timestamp->year % YEARS_IN_LEAP);

      // apply remaining days based on days in months
      int daysInCurrentMonth;

      while(days)
      {
         // get days in current month
         daysInCurrentMonth = daysInMonth[timestamp->month
- 1];
         if(timestamp->month == 2 && leap)
            daysInCurrentMonth = daysInCurrentMonth + 1;

         // days > days in current month
         if(days > daysInCurrentMonth)
         {
            // increment month
            timestamp->month += 1;
            days = days - daysInCurrentMonth;

            // month exceeds months in year
            if(timestamp->month > MONTHS_IN_YEAR)
            {
               // increment year and reset month
               timestamp->year += 1; timestamp->month = 1;

               // are we now on a leap year
               leap = !(timestamp->year % YEARS_IN_LEAP);
            }
         }
         else
         {
            // set day of month to remaioning days
            timestamp->day = days; days = 0;
         }
      }

      // set time values to remaining seconds
      timestamp->hour = hms / SECONDS_IN_HOUR;
      hms = hms % SECONDS_IN_HOUR;

      timestamp->minute = hms / SECONDS_IN_MINUTE;
      timestamp->second = hms % SECONDS_IN_MINUTE;
      return;
   }

//
// copyOutZip
//
// Title      : Copy zip data out of class array
// Parameters : char * - buffer to copy zip string into
//
// Return Value  : int - Length of copy
// Comments      :
//

inline int copyOutZip(char *buffer,char *value,int len =
DEFAULT_ZIP_LEN)
{
   int index      = 0;
   int bufferPos  = 0;

   // add each digit of zip number to buffer inserting delimiter
at 5
   while(value[index] && bufferPos < len)
   {
      if(index == 5)
```

```
        buffer[bufferPos++] = ZIP_DELIMITER;

      buffer[bufferPos++] = value[index++];
   }

   // space fill to the required length
   while(bufferPos < len)
      buffer[bufferPos++] = CHAR_FILL;

   buffer[bufferPos] = NULL;
   return len;
}

//
// copyOutPhone
//
// Title      : Copy phone data out of class array
// Parameters : char * - buffer to copy phone string into
//
// Return Value  : int - Length of copy
// Comments      :
//

inline int copyOutPhone(char *buffer,char *value,int len =
DEFAULT_PHONE_LEN)
{
   int index      = 0;
   int bufferPos  = 0;

   // add each digit of phone number to buffer inserting
delimiter before 6, 9, and 12
   while(value[index] && index < len)
   {
      switch(index)
      {
      case  6:
      case  9:
      case 12:
         // insert delimiter
         buffer[bufferPos++] = PHONE_DELIMITER;
      default:
         // add phone digit to buffer
         buffer[bufferPos++] = value[index++];
      }
   }

   // space fill to the required length
   while(bufferPos < len)
      buffer[bufferPos++] = CHAR_FILL;

   buffer[bufferPos] = '\0';

   return len;
}

//
// copyInMoney64
//
// Title      : Copy money data into class array
// Parameters : const char * - value string
// Return Value  : INT64b integer value
// Comments      :
//

inline bool copyInMoney64(const char * value,INT64b
*number)
{
   //INT64b  number       = 0;
   int      index       = 0;
   int      decimal     = 0;
   int      decimals    = 0;
   int      digitsAfterDec= 0;

   bool   negitiveFlag= false;

   // convert each digit to a numeric portion
   while(value[index])
   {
      // handle $ . - All the rest assumed numeric
      switch(value[index])
      {
      case MONEY_SYMBOL:
         // ignore $ sign
         break;
      case NEGITIVE_SYMBOL:
         // set negitive flag
         negitiveFlag = true;
         break;

      case DECIMAL_SYMBOL:
         // set decimal
         decimal=1;
         decimals++;
         if(decimals >1)
            //more than 1 decimal point found
            return false;
         break;

      default:
         // adjust decimal places
         decimal = decimal * 10;

         // add digit to running total
         if(value[index] >= '0' && value[index] <= '9')
         {
            if(decimal)
               if(++digitsAfterDec > 2)
                  return false;

            *number = (*number * 10) + (value[index] - '0');
         }
         else
         {
            //non-numeric field inserted
            return false;
         }
      }
      index++;
   }


   // apply decimal where decimal not found
   if(decimal < 100)
   {
      if(decimal)
      {
         *number *= (100 / decimal);
      }
      else
      {
         *number *= 100;
      }
   }

   // make negitive
   if(negitiveFlag)
      *number = *number * (-1);

   return true;
}

inline bool copyInMoney32(const char * value,double
*number)
{
   int      index       = 0;
   int      decimal     = 0;
   int      decimals    = 0;
   int      digitsAfterDec= 0;

   bool   negitiveFlag= false;

   // convert each digit to a numeric portion
   while(value[index])
   {
      // handle $ . - All the rest assumed numeric
      switch(value[index])
```

```
    {
    case MONEY_SYMBOL:
       // ignore $ sign
       break;
    case NEGITIVE_SYMBOL:
       // set negitive flag
       negitiveFlag = true;
       break;

    case DECIMAL_SYMBOL:
       // set decimal
       decimal=1;
       decimals++;
       if(decimals >1)
          //more than 1 decimal point found
          return false;
       break;

    default:
       // adjust decimal places
       decimal = decimal * 10;

       // add digit to running total
       if(value[index] >= '0' && value[index] <= '9')
       {
          if(decimal)
             if(++digitsAfterDec > 2)
                return false;

          *number = (*number * 10) + (value[index] - '0');
       }
       else
       {
          //non-numeric field inserted
          return false;
       }
    }
    index++;
 }


 // apply decimal where decimal not found
 if(decimal < 100)
 {
    if(decimal)
    {
       *number *= (100 / decimal);
    }
    else
    {
       *number *= 100;
    }
    }

    // make negitive
    if(negitiveFlag)
       *number = *number * (-1);

    return true;
 }

 //
 // copyInMoney
 //
 // Title      : Convert char string money field to double
 // Parameters : const char * - value string
 // Return Value  : double integer value
 // Comments      :
 //

 inline int copyInMoney(const char *value)
 {
    char buf[20];
    int i,j,decimalFound,digitsAfterDecimal=0;

    int decimal=0;

    //walk past $ if present in char string
    if(*value == '$')
       *value++;

    int len=(int)strlen(value);
    for (i=0;i<len;i++)
    {
       if(value[i] == '.')
       {
          decimalFound++;
          if(decimalFound > 1)
             return -1;
       }
       if(value[i] == '-')


       if (value[i] != '.')
       {
          if(decimal)
          {
             if(digitsAfterDecimal<2)
                digitsAfterDecimal++;
             else
                return -1;
          }
          buf[j++] = value[i];
       }
    }
    int amount = atoi(buf);

    return amount;
 }


 //
 // copyOutMoney64
 //
 // Title      : Copy money data out of class array
 // Parameters : char * - buffer to copy string 64 bit money into
 //              INT64b - value
 //              unsigned len - max number of bytes to copy
 // Return Value  : int - Length of copy
 // Comments      :
 //

 inline void copyOutMoney64(char *buffer,INT64b
 value,unsigned int len = DEFAULT_MONEY64_LEN)
 {
    unsigned intindex       = len;

    int           places        = 0;

    bool          negitiveFlag= false;
    bool          moneyFlag     = true;

    // NULL terminate string
    buffer[index] = NULL;

    // check length > 0
 // if(!index) return len;

    // handle negitive value
    if(value < 0)
    {
       negitiveFlag = true;
       value = value * (-1);
    }

    // break off each digit from value, fill if needed
    do
    {
       if(value)
       {
          // get next digit and add to buffer
          buffer[--index] = (char) (value % 10 + '0');
          value /= 10; places++;

          if(places == 2 && index)
          {
```

```
            places++;
            buffer[--index] = DECIMAL_SYMBOL;
         }
      }
      else
      {
         // add zeros to first place before decimal point on (i.e.
0.00)
         if(places < 2 || places == 3)
         {
            buffer[--index] = ZERO_SYMBOL;
         }
         else
         {
            // add the decimal point
            if(places == 2)
            {
               buffer[--index] = DECIMAL_SYMBOL;
            }
            else
            {
               // add the negitive indicator
               if(negitiveFlag)
               {
                  negitiveFlag = false;
                  buffer[--index] = NEGITIVE_SYMBOL;
               }
               else
               {
                  // add the money indicator
                  if(moneyFlag)
                  {
                     moneyFlag = false;
                     buffer[--index] = MONEY_SYMBOL;
                  }
                  else buffer[--index] = NUMERIC_FILL;
               }
            }
         }

         // need to trace place for decimal point and zero fill
         places++;
      }
   } while(index);

   //return len;
}

//
// copyOutDateTime
//
// Title      : Copy date & time data out of class array
// Parameters : char * - buffer to copy date & time string into
//              INT64b - value
// Return Value  : int - Length of copy
// Comments      : Fixed length
//

inline int copyOutDateTime(char*buffer,INT64b value)
{
   datetime timestamp;

   // break value into time/date components
   calcOutDateTime(value,&timestamp);

   // put month into buffer
   *buffer++ = (char) ((timestamp.month / 10) + '0');
   *buffer++ = (char) ((timestamp.month % 10) + '0');
   *buffer++ = DATE_DELIMITER;

   // put day into buffer
   *buffer++ = (char) ((timestamp.day / 10) + '0');
   *buffer++ = (char) ((timestamp.day % 10) + '0');
   *buffer++ = DATE_DELIMITER;

   // put yearinto buffer
   int   year = timestamp.year;
   *buffer++ = (char) ((year / 1000) + '0'); year = year % 1000;
   *buffer++ = (char) ((year / 100) + '0');  year = year % 100;
   *buffer++ = (char) ((year / 10) + '0');
   *buffer++ = (char) ((year %  10) + '0');
   *buffer++ = CHAR_FILL;

   // put hour   into buffer
   *buffer++ = (char) ((timestamp.hour / 10) +  '0');
   *buffer++ = (char) ((timestamp.hour % 10) +'0');
   *buffer++ = TIME_DELIMITER;

   // put minute into buffer
   *buffer++ = (char) ((timestamp.minute /  10) + '0');
   *buffer++ = (char) ((timestamp.minute %   10) + '0');
   *buffer++ = TIME_DELIMITER;

   // put second into buffer
   *buffer++ = (char) ((timestamp.second / 10) + '0');
   *buffer++ = (char) ((timestamp.second %  10) + '0');

   *buffer  = NULL;   return DEFAULT_DATETIME_LEN;
}
//
// copyOutTime
//
// Title      : Copy date data out of class array
// Parameters : char * - buffer to copy date string into
//              INT64b - value
// Return Value  : int - Length of copy
// Comments      : Fixed length
//

inline int copyOutDate(char *buffer,INT64b value)
{
   datetime timestamp;

   // break value into time/date components
   calcOutDateTime(value,&timestamp);

   // put month into buffer
   *buffer++ = (char) ((timestamp.month / 10) + '0');
   *buffer++ = (char) ((timestamp.month % 10) + '0');
   *buffer++ = DATE_DELIMITER;

   // put day into buffer
   *buffer++ = (char) ((timestamp.day / 10) + '0');
   *buffer++ = (char) ((timestamp.day % 10) + '0');
   *buffer++ = DATE_DELIMITER;

   // put year into buffer
   int year = timestamp.year;
   *buffer++ = (char) ((year / 1000) + '0'); year = year % 1000;
   *buffer++ = (char) ((year / 100) + '0');   year = year % 100;
   *buffer++ = (char) ((year / 10) + '0');
   *buffer++ = (char) ((year % 10) + '0');
   *buffer++ = CHAR_FILL;

   *buffer = NULL;

   return DEFAULT_DATE_LEN;
}

//
// copyOutTime
//
// Title      : Copy time data out of class array
// Parameters : char * - buffer to copy time string into
//              INT64b - value
// Return Value  : int - Length of copy
// Comments      : Fixed length TBD
//

inline int copyOutTime(char *buffer,INT64b value)
{
   datetime timestamp;

   // break value into time/date components
   calcOutDateTime(value,&timestamp);
```

```
   // put hour into buffer
   *buffer++ = (char) ((timestamp.hour / 10) + '0');
   *buffer++ = (char) ((timestamp.hour % 10) + '0');
   *buffer++ = TIME_DELIMITER;

   // put minute into buffer
   *buffer++ = (char) ((timestamp.minute / 10) + '0');
   *buffer++ = (char) ((timestamp.minute % 10) + '0');
   *buffer++ = TIME_DELIMITER;

   // put second into buffer
   *buffer++ = (char) ((timestamp.second / 10) + '0');
   *buffer++ = (char) ((timestamp.second % 10) + '0');

   *buffer = NULL; return DEFAULT_TIME_LEN;
}

//
// copyOutDecimal64
//
// Title      : Copy decimal data out of class array
// Parameters : char * - buffer to copy string 64 bit money into
//              INT64b - value
//              unsigned len - max number of bytes to copy
// Return Value  : int - Length of copy
// Comments      :
//

inline int copyOutDecimal64(char *buffer,INT64b
value,unsigned int len = DEFAULT_DECIMAL64_LEN)
{
   unsigned intindex       = len;

   int           places       = 0;

   bool          negitiveFlag= false;

   // NULL terminate string
   buffer[index] = NULL;

   // check length > 0
   if(!index) return len;

   // handle negitive value
   if(value < 0)
   {
      negitiveFlag = true;
      value = value * (-1);
   }

   // break off each digit from value, fill if needed
   do
   {
      if(value)
      {
         // get next digit and add to buffer
         buffer[--index] = (char) (value % 10 + '0');
         value /= 10; places++;

         if(places == 2 && index)
         {
            places++;
            buffer[--index] = DECIMAL_SYMBOL;
         }
      }
      else
      {
         // add zeros to first place before decimal point on (i.e.
0.00)
         if(places < 2 || places == 3)
         {
            buffer[--index] = ZERO_SYMBOL;
         }
         else
         {
            // add the decimal point
            if(places == 2)
            {
               buffer[--index] = DECIMAL_SYMBOL;
            }
            else
            {
               // add the negitive indicator
               if(negitiveFlag)
               {
                  negitiveFlag = false;
                  buffer[--index] = NEGITIVE_SYMBOL;
               }
               else buffer[--index] = NUMERIC_FILL;
            }
         }

         // need to trace place for decimal point and zero fill
         places++;
      }
   } while(index);

   return len;
}

////////////////////////////////////////////////////////////////////////
//  Macros
////////////////////////////////////////////////////////////////////////
using namespace std;

#ifdef DEBUG
   extern int debugFlag;
#else
   extern int debugFlag;
#endif

inline BYTE8b *debugFileName(BYTE8b *filePath)
{
   BYTE8b *fileName = filePath + strlen(filePath);

    while(fileName != filePath)
    {
      if(*fileName == '/' || *fileName == '\\' && *(fileName + 1))
         return (fileName + 1);

       fileName--;
    }

    return filePath;
}

extern char *get_time_prefix(char *buffer);

#define DEBUGADDRESS(POINTER)hex << (void *)
POINTER << dec

#define DEBUGMSG2(TEXT) ;

#define ERRORMSG(TEXT)   {                                  \
                EnterCriticalSection(&errorMutex);          \
                char buf[50];                               \
                errorStream << debugFileName(__FILE__)
  \
                << "|" << get_time_prefix(buf) << "|" << __LINE__
<< "|" \
                << _getpid() << "|" << GetCurrentThreadId() <<
"|"   \
                << TEXT;                                    \
                errorStream.flush();                        \
                LeaveCriticalSection(&errorMutex);          \
                }

#ifdef DEBUG

#define DEBUGMSG(TEXT)   {                                  \
                EnterCriticalSection(&debugMutex);          \
                char buf[50];                               \
                debugStream << debugFileName(__FILE__)
  \
                << "|" << get_time_prefix(buf) << "|" << __LINE__
<< "|" \
```

```
            << _getpid() << "|" << GetCurrentThreadId() <<
"|"   \
            << TEXT ;                           \
            debugStream.flush();                \
            LeaveCriticalSection(&debugMutex);          \
            }


  #define DEBUGSTRING(TEXT,LENGTH)
  \
      debugVarString(TEXT,LENGTH)


#else
      #define DEBUGMSG(TEXT) ;
      #define DEBUGSTRING(TEXT,LENGTH) ;

#endif
#endif /* _COMMON_TPCC */
```

tpccIsapi.cpp

```
/*
********************************************************************
** Project    : AIX
** Component: Performance/TPC-C Benchmark
** Name        : tpccIsapi.cpp
** Title     : TPCC html processing
********************************************************************
** Copyright (c) 2003 IBM Corporation
** All rights reserved
********************************************************************
** History       :
**              Developed at IBM Austin by the AIX RS/6000
**              performance group.
**
** Comments      :
**
********************************************************************
*/

#include "stdafx.h"

#include "..\tpccCom\tpccCom.h"
#include "..\tpccCom\tpccCom_i.c"
#include "tpccIsapi.hpp"

// For custom assert and trace handling with WebDbg.exe
[ module(name="tpccIsapi", type="dll") ];
[ emitidl(restricted) ];

#define _WIN32_DCOM

#ifdef _DEBUG
   int debugFlag = 1;
#else
   int debugFlag = 0;
#endif

////////////////////////////////////////////////////////////
// Globals
////////////////////////////////////////////////////////////

int     maxDataSize;        //max struct size of all txn(s)
int     numUsers;           //number of users that client will
service.
int     FirstClient;
int     dlvyQueueLen;         //static length of dlvy queue
int     dlvyThreads;        //number of dlvy threads to create
int     dlvyBufferFreeSlots;     //length of dlvy txn queue
int     dlvyBufferSlotIndex;   //index into next avaliable slot in
dlvy txn queue
int     dlvyBufferThreadIndex;    //thread index into dlvy txn
queue
int     nullDB;             //null db on client(bypass com call).
int     trace;

static DWORD       threadLSIndex;       //isapi thread local
storage index
CRITICAL_SECTION   isapiLock;       //isapi lock
CRITICAL_SECTION   errorLock;         //error log file lock.
CRITICAL_SECTION   termLock;        //terminal array lock.
CRITICAL_SECTION   dlvyQueueLock;     //dlvy queue
critical section lock
HANDLE             dlvyThreadDone =
INVALID_HANDLE_VALUE;     //dlvy thread exit event
HANDLE             dlvyThreadSemaphore =
INVALID_HANDLE_VALUE;  //dlvy thread wrk to do
semaphore
int                dlvyThreadID = 0;

struct DLVYQUEUEDATA *dlvyQueue;          //dlvy queue
HANDLE             *dlvyThreadHandles;    //ptr to array of
thread handles

TERM_ENTRY *termArray;                //array of terminal
entries to store each users info.
int        termNextFree;              //next available slot in
terminal array

FILE  *htmlDebug    = NULL;         //html debug file
FILE  *errorLog     = NULL;         //error file
FILE  *htmlTrace    = NULL;

ofstream debugStream;
ofstream errorStream;
CRITICAL_SECTION debugMutex;
CRITICAL_SECTION errorMutex;

char   dlvyLogPath[128]    = {NULL};
char   errorLogFile[128]    = {NULL};
char   htmlTraceLogFile[128]   = {NULL};
char   dbName[64]           = {NULL};
char   dbType[16]          = {NULL};

typedef INT (*CONNECT_PTR)(char *dbName,void
**connectHandle);
typedef INT (*DISCONNECT_PTR)(void *connectHandle);
typedef INT (*DLVY_FUNC_PTR)(dlvy_wrapper *dlvy,void
*connectHandle);
typedef INT (*NORD_FUNC_PTR)(nord_wrapper *nord,void
*connectHandle);
typedef INT (*PYMT_FUNC_PTR)(paym_wrapper *pymt,void
*connnectHandle);
typedef INT (*ORDS_FUNC_PTR)(ords_wrapper *ords,void
*connectHandle);
typedef INT (*STOK_FUNC_PTR)(stok_wrapper *stok,void
*connectHandle);

HINSTANCE     dbInstance;
CONNECT_PTR     db_connect;
DISCONNECT_PTR   db_disconnect;
DLVY_FUNC_PTR  dlvyCall;


////////////////////////////////////////////////////////////
// Page functions arrays
////////////////////////////////////////////////////////////

typedef int   (*pageFuncPtr) (htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);

pageFuncPtr htmlPageFunctions[MAX_TRANSACTIONS] =
        {
          {doLoginForm},
          {doNewOrderForm},
          {doPaymentForm},
          {doOrderStatusForm},
          {doDeliveryForm},
          {doStockForm},
          {doExit},
          {doLoginResults},
          {doNewOrderResults},
          {doPaymentResults},
          {doOrderStatusResults},
          {doDeliveryResults},
```

```
        {doStockResults}
      };

extern "C" DWORD WINAPI
HttpExtensionProc(LPEXTENSION_CONTROL_BLOCK
lpECB)
{
  struct TXN_HANDLE  *txnHandle = NULL;

  txnHandle = (TXN_HANDLE *)
TlsGetValue(threadLSIndex);

  if(txnHandle == NULL)
  {
    int rc = initTxnHandle(&txnHandle);
    if (rc != OK)
    {
      char response[256]; char htmlHeader[256];
      sprintf(response,"ERROR : Init txnHandle function
failed.\n");

      size_t htmlPageLen = strlen(response);

      //add content length and keep alive header
      sprintf(htmlHeader,HEADER,htmlPageLen);
      lpECB->ServerSupportFunction(lpECB-
>ConnID,HSE_REQ_SEND_RESPONSE_HEADER,"200
OK",NULL,(DWORD*)htmlHeader);
      lpECB->WriteClient(lpECB-
>ConnID,response,(LPDWORD)&htmlPageLen,0);

      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }

    txnHandle = (TXN_HANDLE *)
TlsGetValue(threadLSIndex);
    if (txnHandle == NULL)
    {
      char response[256]; char htmlHeader[256];
      sprintf(response,"ERROR : Unable to retrieve
txnHandle from TLS.\n");

      size_t htmlPageLen = strlen(response);

      //add content length and keep alive header
      sprintf(htmlHeader,HEADER,htmlPageLen);
      lpECB->ServerSupportFunction(lpECB-
>ConnID,HSE_REQ_SEND_RESPONSE_HEADER,"200
OK",NULL,(DWORD*)htmlHeader);
      lpECB->WriteClient(lpECB-
>ConnID,response,(LPDWORD)&htmlPageLen,0);

      return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }
  }

  try
  {
    txnHandle->urlString  = (char*)lpECB->lpszQueryString;

    DEBUGMSG("calling doHtml() w/ query string:" <<
txnHandle->urlString << endl);
    doHtml(txnHandle);

    size_t htmlPageLen;
    htmlPageLen = strlen(txnHandle->htmlPage);
    if(htmlPageLen >= 4096)
    {
      ERRORMSG("WARNING:  HTML PAGE IS >= 4096!,
page size:"<<htmlPageLen<<endl);
    }
    //add content length and keep alive header
    sprintf(txnHandle->htmlHeader,HEADER,htmlPageLen);
    size_t headerLen = strlen(txnHandle->htmlHeader);
    if(headerLen >= 256)
    {
      ERRORMSG("WARNING:  HTML HEADER IS >= 256!,
header size:"<<headerLen<<endl);
    }

    //write response to user
    lpECB->ServerSupportFunction(lpECB-
>ConnID,HSE_REQ_SEND_RESPONSE_HEADER,"200
OK",NULL,(DWORD*)txnHandle->htmlHeader);
    lpECB->WriteClient(lpECB->ConnID,txnHandle-
>htmlPage,(LPDWORD)&htmlPageLen,0);

    DEBUGMSG("HTML PAGE-->"<<endl<<txnHandle-
>htmlHeader<<txnHandle->htmlPage<<endl);

  }
  catch (...)
  {
    char response[256];
    ZeroMemory(response,256);
    char *ptr = response;

    appendText(&ptr,"<HTML><BODY> Error : Unhandled
Exception </BODY></HTML>");
    DWORD   cbResponse = sizeof(response)-1 ;

    //write response to user
    lpECB->ServerSupportFunction(lpECB-
>ConnID,HSE_REQ_SEND_RESPONSE_HEADER,"200
OK",NULL,(DWORD*)response);
    lpECB->WriteClient(lpECB-
>ConnID,response,&cbResponse,0);
  }

   return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}

extern "C" BOOL WINAPI
GetExtensionVersion(HSE_VERSION_INFO* pVer)
{
  // Create the extension version string, and copy string to
HSE_VERSION_INFO structure.
  pVer->dwExtensionVersion =
MAKELONG(HSE_VERSION_MINOR,
HSE_VERSION_MAJOR);

  // Copy description string into HSE_VERSION_INFO
structure.
 strcpy(pVer->lpszExtensionDesc, "TPCC ISAPI Extension");

  // Initialze isapi critcal section
  InitializeCriticalSection(&isapiLock);

  // Initailize error log critcal section
  InitializeCriticalSection(&errorLock);

  // Initialze terminal critcal section
  InitializeCriticalSection(&termLock);

  // Initialize debug/error critcal sections
  if(debugFlag)
    InitializeCriticalSection(&debugMutex);
  InitializeCriticalSection(&errorMutex);

  // Read registery values
  if(readRegisteryValues() != OK)
    return(FALSE);

  // Initialize terminal array
  termArray = (TERM_ENTRY*)
calloc(numUsers,sizeof(TERM_ENTRY));
  termNextFree = 1;

  //open up error/debug streams
  errorStream.rdbuf( )->open(errorLogFile,ios::out);
  if(debugFlag)
    debugStream.rdbuf( )->open(htmlTraceLogFile,ios::out);

  ERRORMSG("Error log file open."<<endl);
```

```
   DEBUGMSG("Loading library for dlvy txn."<<endl);
   int rc = getDBInstance();
   if (rc != OK)
   {
      ERRORMSG("Error, unable to load database dll,
rc:"<<rc);
      DEBUGMSG("Error, unable to load database dll,
rc:"<<rc);

      return FALSE;
   }
   DEBUGMSG("Library loaded for dlvy txn."<<endl);

   DEBUGMSG("Calling initDlvy." <<endl);
   ERRORMSG("Calling initDlvy." <<endl);

   if(initDlvy() != OK)
      return (FALSE);

   DEBUGMSG("Initializing TLS." << endl);

   // Initialize thread local storage index
   threadLSIndex = TlsAlloc();
   if (threadLSIndex == TLS_NULL)
   {
      ERRORMSG("Isapi error: unable to initialize thread local
storage(TLS), rc:" << GetLastError()<<endl);
      return(FALSE);
   }
   ERRORMSG("Initialized TLS." << endl);

   DEBUGMSG("sizeof out_neword_struct:   "<<sizeof(struct
out_neword_type)<<endl);
   DEBUGMSG("sizeof  in_neword_struct:   "<<sizeof(struct
in_neword_type)<<endl);
   DEBUGMSG("sizeof out_payment_struct:  "<<sizeof(struct
out_payment_struct)<<endl);
   DEBUGMSG("sizeof  in_payment_struct:  "<<sizeof(struct
in_payment_struct)<<endl);
   DEBUGMSG("sizeof out_ordstat_struct:  "<<sizeof(struct
out_ordstat_struct)<<endl);
   DEBUGMSG("sizeof  in_ordstat_struct:  "<<sizeof(struct
in_ordstat_struct)<<endl);
   DEBUGMSG("sizeof out_delivery_struct: "<<sizeof(struct
out_delivery_struct)<<endl);
   DEBUGMSG("sizeof  in_delivery_struct: "<<sizeof(struct
in_delivery_struct)<<endl);
   DEBUGMSG("sizeof out_stocklev_struct: "<<sizeof(struct
out_stocklev_struct)<<endl);
   DEBUGMSG("sizeof  in_stocklev_struct: "<<sizeof(struct
in_stocklev_struct)<<endl);

   //compute the max struct size for com data construct
   maxDataSize = max(maxDataSize,sizeof(nord_wrapper));
   maxDataSize = max(maxDataSize,sizeof(paym_wrapper));
   maxDataSize = max(maxDataSize,sizeof(ords_wrapper));
   maxDataSize = max(maxDataSize,sizeof(dlvy_wrapper));
   maxDataSize = max(maxDataSize,sizeof(stok_wrapper));
   maxDataSize += 10;

   DEBUGMSG("max data struct size:"<<maxDataSize
<<endl);
   ERRORMSG("max data struct size:"<<maxDataSize
<<endl);

   return true;
}

extern "C" BOOL WINAPI TerminateExtension(DWORD
dwFlags)
{
// ERRORMSG("TerminateExtension"<<endl);
   return true;
}


/*
**********************************************************************
** Name         :  initTxnHandle
** Description :
**             Isapi thread initializes its own com interface
**             structure.
** Parameters:
**             TXN_HANDLE*    isapi txn handle
** Returns      :
**             int - return code
** Comments     :
**
**********************************************************************
*/
int initTxnHandle(TXN_HANDLE **txnHandle)
{
   DEBUGMSG("Inside init txn handle, getting isapiLock." <<
endl);
// ERRORMSG("Inside init txn handle, getting isapiLock." <<
endl);

   EnterCriticalSection(&isapiLock);

   HRESULT hres = NULL;

   try
   {
      DEBUGMSG("Got ispaiLock, initializing txnHandle:
"<<DEBUGADDRESS(*txnHandle)<< endl);
//    ERRORMSG("Got ispaiLock, initializing txnHandle:
"<<DEBUGADDRESS(*txnHandle)<< endl);
      *txnHandle = (TXN_HANDLE *)
calloc(1,sizeof(TXN_HANDLE));
      if (*txnHandle == NULL)
      {
         ERRORMSG("Unable to allocated TXN_HANDLE,
rc:"<<GetLastError()<<endl);
         return ERR;
      };

      (*txnHandle)->comInterface.comHandle = NULL;
      DEBUGMSG("Initializing txnHandle com data buffer to
"<<maxDataSize<<"bytes"<<endl);
//    ERRORMSG("Initializing txnHandle com data buffer to
"<<maxDataSize<<"bytes"<<endl);
      (*txnHandle)->comInterface.txnBuffer = (char *)
CoTaskMemAlloc(maxDataSize);
      if (!((*txnHandle)->comInterface.txnBuffer))
      {
         ERRORMSG("CoTaskMemAlloc() failed of size
"<<maxDataSize<<", rc: "<<hres<<endl);
         return(ERR);
      };
      DEBUGMSG("txnHandle com data buffer initialized to "
<< maxDataSize << "bytes" <<endl);
//    ERRORMSG("txnHandle com data buffer initialized to "
<< maxDataSize << "bytes" <<endl);

      //(*txnHandle)->comInterface.comHandle = NULL;
      DEBUGMSG("Calling CoInitialize with txnHandle:
"<<DEBUGADDRESS(*txnHandle)<<endl);
//    ERRORMSG("Calling CoInitialize with txnHandle:
"<<DEBUGADDRESS(*txnHandle)<<endl);
      hres = CoInitializeEx(NULL,COINIT_MULTITHREADED);
      if (FAILED(hres))
      {
         ERRORMSG("CoInitializeEx() failed, rc :
"<<hres<<endl);
         return(ERR);
      };

      struct  _timeb          startTime;
      struct  _timeb          endTime;

      DEBUGMSG("Calling CoCreateInstance with
txnHandle:"<<DEBUGADDRESS(*txnHandle)<< endl);
//    ERRORMSG("Calling CoCreateInstance with
txnHandle:"<<DEBUGADDRESS(*txnHandle)<< endl);
      _ftime(&startTime);
```

```
    hres =
CoCreateInstance(CLSID_tpcc_com,NULL,CLSCTX_SERVE
R,IID_Itpcc_com,(void **)&(*txnHandle)-
>comInterface.comHandle);
    if (FAILED(hres))
    {
       _ftime(&endTime);
       //store error code in txnHandle
       ERRORMSG("CoCreateInstance() failed,
code:"<<HRESULT_CODE(hres)<<"
facility:"<<HRESULT_FACILITY(hres)<<
          " hres:"<<hres<< " time waiting:"<<
          ((((endTime.time - startTime.time)*1000)+
            (endTime.millitm -
startTime.millitm))/1000.0)<<endl);

       DEBUGMSG("CoCreateInstance() failed,
code:"<<HRESULT_CODE(hres)<<"
facility:"<<HRESULT_FACILITY(hres)<<
          " hres:"<<hres<< " time waiting:"<<
          ((((endTime.time - startTime.time)*1000)+
            (endTime.millitm -
startTime.millitm))/1000.0)<<endl);

       return(ERR);
    };

    _ftime(&endTime);
    DEBUGMSG("CoCreateInstance successful.txnHande
com initialized, time waiting for object to be activated:" <<
         ((((endTime.time - startTime.time)*1000)+
           (endTime.millitm -
startTime.millitm))/1000.0)<<endl);
//    ERRORMSG("CoCreateInstance successful.txnHande
com initialized, time waiting for object to be activated:" <<
//         ((((endTime.time - startTime.time)*1000)+
//           (endTime.millitm -
startTime.millitm))/1000.0)<<endl);

    //call set complete to return object to pool.
    (*txnHandle)->comInterface.comHandle-
>doSetComplete();

    hres = (*txnHandle)->comInterface.comHandle-
>doDBInfo();
    (*txnHandle)->comInterface.comHandle-
>doSetComplete();

    //set the com buffers size
    DEBUGMSG("Setting txnHandle: " <<
DEBUGADDRESS(*txnHandle) << "com buffer size to " <<
maxDataSize<< endl)
//    ERRORMSG("Setting txnHandle: " <<
DEBUGADDRESS(*txnHandle) << "com buffer size to " <<
maxDataSize<< endl)
    (*txnHandle)->comInterface.size = maxDataSize;

    DEBUGMSG("txnHandle:
"<<DEBUGADDRESS(*txnHandle) <<"set to " <<
maxDataSize << endl);
//    ERRORMSG("txnHandle:
"<<DEBUGADDRESS(*txnHandle) <<"set to " <<
maxDataSize << endl);

    TlsSetValue(threadLSIndex,*txnHandle);

    DEBUGMSG("txnHandle:
"<<DEBUGADDRESS(*txnHandle) << "stored in TLS" <<
endl);
//    ERRORMSG("txnHandle:
"<<DEBUGADDRESS(*txnHandle) << "stored in TLS" <<
endl);

    ZeroMemory((*txnHandle)-
>htmlPage,MAX_HTML_PAGE_LEN);
    ZeroMemory((*txnHandle)-
>htmlHeader,MAX_HTML_HEADER_LEN);

    LeaveCriticalSection(&isapiLock);
    return(OK);
  }
  catch(...)
  {
    DEBUGMSG("Unhandled exeception in initTxnHandle,
unlocking isapi lock" <<endl);
//    ERRORMSG("Unhandled exeception in initTxnHandle,
unlocking isapi lock" <<endl);
    LeaveCriticalSection(&isapiLock);
  };

  return ERR;
}

/*
**********************************************************************
** Name        :  getDBInstance
** Description :
**            load db specific lib based on dbType registery
**            value.
** Parameters:
**
** Returns     :
**            int - return code
** Comments    :
**            This function only exists for the dlvy threads
**            Dlvy threads hold direct connections to the
database
**            and therefore need to know what db interface to
talk to.
**********************************************************************
*/
int getDBInstance()
{
  if(nullDB)
  {
    dbInstance =
LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\nullDB.dll");
    if(dbInstance == NULL)
    {
      return ERR_NULL_DLL_NOT_LOADED;
    }
  }
  else if( (strcmp(dbType,"DB2") == 0) )
  {
    dbInstance =
LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccDB2glue.dll");
    if(dbInstance == NULL)
    {
      return ERR_DB2_DLL_NOT_LOADED;
    }
  }
  else if( (strcmp(dbType,"ORACLE") == 0) )
  {
    ERRORMSG("Loading Oracle dll"<<endl);
    dbInstance =
LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccOracleGlue.dll");
    if(dbInstance == NULL)
    {
      ERRORMSG("Could not Load Oracle dll"<<endl);
      return ERR_ORA_DLL_NOT_LOADED;
    }
    ERRORMSG("Loaded Oracle dll"<<endl);
  }
  else if( (strcmp(dbType,"SYBASE") == 0) )
  {
    ERRORMSG("Loading Sybase dll"<<endl);
    dbInstance =
LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccSybaseGlue_2.d
ll");
    if(dbInstance == NULL)
    {
      ERRORMSG("Could not Load Sybase dll
"<<dbInstance<<endl);
      return ERR_SYB_DLL_NOT_LOADED;
    }
    ERRORMSG("Loaded Sybase dll"<<endl);
```

```
  }
  else
  {
    return ERR_UNKNOWN_DB;
  }

  ERRORMSG("Get address"<<endl);
  db_connect =
(CONNECT_PTR)GetProcAddress(dbInstance,"connect_db");
  if(db_connect == NULL)
  {
    ERRORMSG("Could not find connect_db function in
dll"<<endl);
    return ERR_CONNECT_ADDRESS_NOT_FOUND;
  }
  ERRORMSG("Get address"<<endl);
  dlvyCall =
(DLVY_FUNC_PTR)GetProcAddress(dbInstance,"do_dlvy");
  if(dlvyCall == NULL)
  {
    ERRORMSG("Could not find do_dlvy in dll"<<endl);
    return ERR_DLVY_ADDRESS_NOT_FOUND;
  }
  return OK;
}

/*
*********************************************************************
** Name         :  initDlvy
** Description :
**             initialize dlvy threads/dlvy queueu
** Parameters:
**
** Returns      :
**             int - return code
** Comments     :
**
*********************************************************************
*/

int initDlvy()
{
  ERRORMSG(">>initDlvy"<<endl);
  // Initialze critcal section
  InitializeCriticalSection(&dlvyQueueLock);

  //create dlvy queue
  dlvyQueue = (DLVYQUEUEDATA *)
calloc(dlvyQueueLen,sizeof(DLVYQUEUEDATA));

  if (dlvyQueue == NULL)
  {
    ERRORMSG("calloc failed to allocate dlvyQueue"<<endl);
    return  ERR_DLVY_QUEUE_CALLOC_FAIL;
  }
  ERRORMSG(">>calloc"<<endl);
  //init dlvy buffer critcal section
  //InitializeCriticalSection(&dlvyQueueLock);

  dlvyThreadDone = CreateEvent(NULL,
                    TRUE,   //manual reset
                    FALSE,  //initially not signalled.
                    NULL);
  if(dlvyThreadDone == NULL)
  {
    DEBUGMSG("Error: dlvyThreadDone handled init failed,
GetLastError:"<<GetLastError()<<endl);

    ERRORMSG("Error : dlvyThreadDone handled init failed,
GetLastError:"<<GetLastError()<<endl);

    return  ERR_DLVY_EVENT_INIT_FAILED;
  }
  //create dlvy semaphore
  dlvyThreadSemaphore =
CreateSemaphore(NULL,0,dlvyQueueLen,NULL);
  if(dlvyThreadSemaphore == NULL)
  {
    DEBUGMSG("Error: dlvyThreadSemaphore semaphore
init failed, GetLastError:"<<GetLastError()<<endl);
    ERRORMSG("Error: dlvyThreadSemaphore semaphore
init failed, GetLastError:"<<GetLastError()<<endl);
    return  ERR_DLVY_SEMAPHORE_INIT_FAILED;

  }

  //set number of free slots available in queue
  dlvyBufferFreeSlots = dlvyQueueLen;

  //index into next avaliable slot in dlvy txn queue
  dlvyBufferSlotIndex = 0;

  //thread index into dlvy txn queue
  dlvyBufferThreadIndex = 0;

  dlvyThreadHandles = new HANDLE[dlvyThreads];

  //create threads
  for(int threadCount = 0;threadCount <
dlvyThreads;threadCount++)
  {
    ERRORMSG(">>Calling dlvyThreadEntry"<<endl);
    dlvyThreadHandles[threadCount] =
(HANDLE)_beginthread(dlvyThreadEntry,0,NULL);
    if(dlvyThreadHandles[threadCount] ==
INVALID_HANDLE_VALUE) {
      ERRORMSG(">>Calling dlvyThreadEntry
failed"<<endl);
      return ERR_DLVY_THREAD_FAILED;
    }
  }


  return OK;
}


/*
*********************************************************************
** Name         :  readRegisteryValues
** Description :
**             initialize isapi global variables from registery
** Parameters:
**
** Returns      :
**             int - return code
** Comments     :
**
*********************************************************************
*/
int readRegisteryValues()
{
  HKEY   registeryKey;
  char   value[MAX_STRING_LEN];
  DWORD  regType;
  DWORD  regValue;
  DWORD  regValueSize = MAX_STRING_LEN;

// ERRORMSG(">>readRegisteryValues"<<endl);
  //open up registery key
  if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTERY
_SUB_KEY,0,KEY_READ,&registeryKey) !=
ERROR_SUCCESS)
    return ERR_UNABLE_TO_OPEN_REG;

  //get null db flag
  regValueSize = sizeof(regValue);
  if(RegQueryValueEx(registeryKey,NULL_DB,0,&regType,(
BYTE *)&regValue,&regValueSize) == ERROR_SUCCESS)
    nullDB = regValue;
  else
    nullDB = 0;

  //get num dlvy threads
  regValueSize = sizeof(regValue);
```

```
   if(RegQueryValueEx(registeryKey,DELIVERY_THREADS,0
,&regType,(BYTE *)&regValue,&regValueSize) ==
ERROR_SUCCESS)
        dlvyThreads = regValue;
   else
        dlvyThreads = DEFAULT_DLVY_THREADS;

   //get dlvy queue len
   regValueSize = sizeof(regValue);
   if(RegQueryValueEx(registeryKey,DELIVERY_QUEUE_LE
N,0,&regType,(BYTE *)&regValue,&regValueSize) ==
ERROR_SUCCESS)
        dlvyQueueLen = regValue;
   else
        dlvyQueueLen = DEFAULT_DLVY_QUEUE_LEN;

   //get the htmlTrace flag
   regValueSize = sizeof(regValue);
   if(RegQueryValueEx(registeryKey,HTML_TRACE,0,&regTy
pe,(BYTE *)&regValue,&regValueSize) ==
ERROR_SUCCESS)
        trace = regValue;
   else
        trace = 0;

   //get the client null db flag
   regValueSize = sizeof(regValue);
   if(RegQueryValueEx(registeryKey,NULL_DB,0,&regType,(
BYTE *)&regValue,&regValueSize) == ERROR_SUCCESS)
        nullDB = regValue;
   else
        nullDB = 0;

   //get the num of users
   regValueSize = sizeof(regValue);
   if(RegQueryValueEx(registeryKey,NUM_USERS,0,&regTy
pe,(BYTE *)&regValue,&regValueSize) ==
ERROR_SUCCESS)
        numUsers = regValue;
   else
        numUsers = DEFAULT_NUM_USERS;

   //get dlvy log file path
   regValueSize = sizeof(value);
   if
(RegQueryValueEx(registeryKey,DELIVERY_LOG_PATH,0,
&regType,(BYTE *) &value,&regValueSize)==
ERROR_SUCCESS )
      strcpy(dlvyLogPath,value);
   else
      strcpy(dlvyLogPath,DEFAULT_DLVY_LOG_PATH);

   //get global error log file path/name
   regValueSize = sizeof(value);
   if
(RegQueryValueEx(registeryKey,ERROR_LOG_FILE,0,&reg
Type,(BYTE *) &value,&regValueSize)==
ERROR_SUCCESS )
      strcpy(errorLogFile,value);
   else
      strcpy(errorLogFile,DEFAULT_ERROR_LOG_FILE);

   //get global error log file path/name
   regValueSize = sizeof(value);
   if
(RegQueryValueEx(registeryKey,HTML_TRACE_LOG_FILE,
0,&regType,(BYTE *) &value,&regValueSize)==
ERROR_SUCCESS )
      strcpy(htmlTraceLogFile,value);
   else

   strcpy(htmlTraceLogFile,DEFAULT_HTML_TRACE_LOG_
FILE);

   //get db name
   regValueSize = sizeof(value);
   if
(RegQueryValueEx(registeryKey,DB_NAME,0,&regType,(BY
TE *) &value,&regValueSize)== ERROR_SUCCESS )
      strcpy(dbName,value);
   else
      strcpy(dbName,DEFAULT_DB_NAME);

   //get db type
   regValueSize = sizeof(value);
   if
(RegQueryValueEx(registeryKey,DB_TYPE,0,&regType,(BY
TE *) &value,&regValueSize)== ERROR_SUCCESS )
      strcpy(dbType,value);

   //get First Client
   regValueSize = sizeof(regValue);
   if(RegQueryValueEx(registeryKey,"FirstClient",0,&regType,
(BYTE *)&regValue,&regValueSize) == ERROR_SUCCESS)
        FirstClient = regValue;
   else
        FirstClient = 1;

   RegCloseKey(registeryKey);

   return OK;
}

/*
*****************************************************************
** Name        : doLoginForm
** Description :
**          HTML Login page entry point
** Parameters:
**          htmlPhraser command block
**          char *     html result page
** Returns     :
**          int - return code
** Comments    :
**
*****************************************************************
*/

int doLoginForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
   char buffer[20];
   DEBUGMSG("Entering doLoginForm()."<<endl);
// ERRORMSG("Entering doLoginForm()."<<endl);
   char *html=txnHandle->htmlPage;

   DEBUGMSG("Creating html login page"<<endl);
   //begin html page
   appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Client Home Page</TITLE></HEAD>"
              "<FORM ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">"
              "<H2>Please Login.</H2>"
              "<INPUT TYPE=\"hidden\" NAME=\""
              CMD_TXN_ID
              "\" VALUE=\""
              CMD_MENU
              "\">"
              "<H3>Warehouse <INPUT NAME=\""
              CMD_W_ID
              "\" SIZE=6>"
              " District <INPUT NAME=\""
              CMD_D_ID
              "\" SIZE=2></H3>"
              "<INPUT TYPE=\"submit\" VALUE=\"Submit\">"
              "</FORM></BODY></HTML>");

   appendText(&html," dlvyBufferFreeSlots ");
   appendText(&html,itoa(dlvyBufferFreeSlots,buffer,10));
   appendText(&html," dlvyBufferSlotIndex ");
   appendText(&html,itoa(dlvyBufferSlotIndex,buffer,10));
   appendText(&html," dlvyBufferThreadIndex ");
   appendText(&html,itoa(dlvyBufferThreadIndex,buffer,10));

   DEBUGMSG("Html login page done"<<endl);
```

```
   return OK;
}

/*
**********************************************************************
** Name         : doLoginResults
** Description :
**           HTML Login results page entry point
** Parameters:
**           htmlPhraser command block
**           char *     html result page
** Returns      :
**           int - return code
** Comments     :
**
**********************************************************************
*/

int doLoginResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
   char *html=txnHandle->htmlPage;

   //validate parameters
   if( (txnHandle->w_id = atoi(commandBlock->get_W_ID()))
== 0 )
   {
      doLoginErrorPage(html,ERR_INVALID_W_ID);
      return OK;
   }
   if( (txnHandle->d_id = atoi(commandBlock->get_D_ID()))
== 0 )
   {
      doLoginErrorPage(html,ERR_INVALID_D_ID);
      return OK;
   }

   //store user into terminal array,
   //function will ERR if the terminal array is full
   if( assignTerminal(txnHandle) != OK)
   {
      doLoginErrorPage(html,ERR_TERMINAL_FULL);
      return OK;
   };

   appendText(&html,"<HTML><HEAD><TITLE>TPC-C Main
Menu</TITLE></HEAD>\r\n"
               "<BODY><FORM ACTION=\""
               APP_NAME
               "\" METHOD=\"GET\">\r\n"
               "<H3>Please Select Transaction.</H3>\r\n");
   html+=appendButtons(html);

   html+=appendHiddenFields(html,txnHandle);

   appendText(&html,"</FORM></BODY></HTML>");

   return OK;
}

/*
**********************************************************************
** Name         : doLoginErrorPage
** Description :
**           HTML Login page entry point
** Parameters:
**           char *     html page buffer
**           char *     error message
** Returns      :
**           int - return code
** Comments     :
**
**********************************************************************
*/

int doLoginErrorPage(char *htmlPage,char *errorMessage)
{
   char *html=htmlPage;

   //begin html page
   appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Client Home Page</TITLE></HEAD>"
               "<FORM ACTION=\""
               APP_NAME
               "\" METHOD=\"GET\">");

   appendText(&html,"<H2>Please Login.</H2>"
               "<INPUT TYPE=\"hidden\" NAME=\""
               CMD_TXN_ID
               "\" VALUE=\""
               CMD_MENU
               "\">"
               "<H3>Warehouse <INPUT NAME=\""
               CMD_W_ID
               "\" SIZE=6>"
               " District <INPUT NAME=\""
               CMD_D_ID
               "\" SIZE=2></H3>"
               "<INPUT TYPE=\"submit\" VALUE=\"Submit\">"
               "</FORM>");
   appendText(&html,errorMessage);
   appendText(&html,"<BODY></HTML>");

   return OK;
}


/*
**********************************************************************
** Name         : doNewOrderForm
** Description :
**           HTML neworder page entry point
** Parameters:
**           htmlPhraser command block
**           char *     html result page
** Returns      :
**           int - return code
** Comments     :
**
**********************************************************************
*/


int doNewOrderForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
   char *html=txnHandle->htmlPage;

   appendText(&html,"<HTML><HEAD><TITLE>TPC-C New
Order</TITLE></HEAD>\r\n"
               "<BODY><FORM ACTION=\""
               APP_NAME
               "\" METHOD=\"GET\">\r\n"
               "<CENTER><H3>Please Fill In New Order
Form.</H3></CENTER>\r\n" //check if not needed
               "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
               CMD_TXN_ID
               "\" VALUE=\""
               CMD_NORD
               "\">");

   //append the hidden
   html+=appendHiddenFields(html,txnHandle);

   //int buffer for warehouse
   char buffer[15];
   appendText(&html," <PRE>"
//             "       1       2       3       4       5       6       7
8       9\r\n"
//
"12345678901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890\r\n"
               "Warehouse: ");
   appendText(&html,itoa(txnHandle->w_id,buffer,10),7,1);
   appendText(&html,"District: <INPUT NAME=\""
```

```
            CMD_D_ID
            "\" SIZE=1>                    Date:<BR>"
              "Customer <INPUT NAME=\""
            CMD_C_ID
            "\" SIZE=6>  Name:                 Credit:
%Disc.:<BR>"
            "Order Number:          Number of Lines:
W_tax:       D_tax:<BR> <BR>"
//          "       1       2       3       4       5       6       7
8       9\r\n"
//
"12345678901234567890123456789012345678901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890\r\n"
            " Supp_W  Item_Num  Item_Name              Qty
Stock  B/G  Price    Amount <BR> ");

  //append the 15 items commands
  html+=appendItems(html,NORD_ITEMS,ITEM_START);

  //seal up html page
  appendText(&html,"</PRE></BODY></HTML>");

  return OK;
}

/*
**********************************************************************
** Name        : doNewOrderResults
** Description :
**          HTML neworder page entry point
** Parameters:
**          htmlPhraser command block
**          char *     html result page
** Returns     :
**          int - return code
** Comments    :
**
**********************************************************************
*/


int doNewOrderResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
  DEBUGMSG("Entered doNewOrderResults" << endl);

  char *html=txnHandle->htmlPage;
  struct nord_wrapper *nord = NULL;

  DEBUGMSG("Casting COM txnBuffer to nord struct"
<<endl);
  nord = (nord_wrapper*)txnHandle->comInterface.txnBuffer;
  ZeroMemory(nord,sizeof(nord_wrapper));
  DEBUGMSG("COM txnBuffer initialized, validating input
parameters" << endl);

  //set warehouse,district and customer id from command
block
  nord->in_nord.s_W_ID = txnHandle->w_id;
  DEBUGMSG("nord w_id:" <<  nord->in_nord.s_W_ID <<
endl);

  if( (nord->in_nord.s_D_ID = atoi(commandBlock-
>get_D_ID()) ) == 0)
  {

  doNewOrderErrorPage(html,ERR_INVALID_D_ID,comman
dBlock,txnHandle);
    return OK;
  }
  DEBUGMSG("nord d_id:" <<  nord->in_nord.s_D_ID <<
endl);

  if((nord->in_nord.s_C_ID = atoi(commandBlock-
>get_C_ID()) ) == 0)
  {

  doNewOrderErrorPage(html,ERR_INVALID_C_ID,comman
dBlock,txnHandle);
    return OK;
  }
  DEBUGMSG("nord c_id:" <<  nord->in_nord.s_C_ID <<
endl);

  int itemCmd      = ITEM_START;
  short itemComplete = 0;
  char field[256] ={NULL};

 for (int itemIndex=0;itemIndex<NORD_ITEMS;itemIndex++)
  {
    //supply warehouse
    if( *(commandBlock->get_ITEM_SUPP_W(itemIndex)) )
#ifdef DB2
      if ( (nord->in_nord.in_item[nord-
>in_nord.s_O_OL_CNT].s_OL_SUPPLY_W_ID =
atoi(commandBlock->get_ITEM_SUPP_W(itemIndex))) == 0)
#elif ORACLE
      if ( (nord->in_nord.s_OL_SUPPLY_W_ID[nord-
>in_nord.s_O_OL_CNT] = atoi(commandBlock-
>get_ITEM_SUPP_W(itemIndex))) == 0)
#elif SYBASE
      if ( (nord->in_nord.s_OL_SUPPLY_W_ID[nord-
>in_nord.s_O_OL_CNT] = atoi(commandBlock-
>get_ITEM_SUPP_W(itemIndex))) == 0)
#endif
      {

  doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_I
D,commandBlock,txnHandle);
        return OK;
      }
      else
        itemComplete++;

    //item number
    if( *(commandBlock->get_ITEM_ITEM_NUM(itemIndex)) )
    {
      if(itemComplete==1)
      {
#ifdef DB2
        if ( (nord->in_nord.in_item[nord-
>in_nord.s_O_OL_CNT].s_OL_I_ID = atoi(commandBlock-
>get_ITEM_ITEM_NUM(itemIndex))) == 0)
#elif ORACLE
        if ( (nord->in_nord.s_OL_I_ID[nord-
>in_nord.s_O_OL_CNT] = atoi(commandBlock-
>get_ITEM_ITEM_NUM(itemIndex))) == 0)
#elif SYBASE
        if ( (nord->in_nord.s_OL_I_ID[nord-
>in_nord.s_O_OL_CNT] = atoi(commandBlock-
>get_ITEM_ITEM_NUM(itemIndex))) == 0)
#endif
        {

  doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,co
mmandBlock,txnHandle);
          return OK;
        }
        else
          itemComplete++;
      }
      //missing previous value of item supp warehouse, flag
error
      else
      {

  doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_I
D,commandBlock,txnHandle);
        return OK;
      }
    }
    else if( (itemComplete==1) ) //nothing in the command
block, check to see if the previous item value is present
    {
```

```
   doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,co
mmandBlock,txnHandle);
        return OK;
      }

      //item qty
      if(*(commandBlock->get_ITEM_QTY(itemIndex)))
      {
        if(itemComplete==2)
        {
#ifdef DB2
          if( (nord->in_nord.in_item[nord-
>in_nord.s_O_OL_CNT].s_OL_QUANTITY =
atoi(commandBlock->get_ITEM_QTY(itemIndex))) == 0)
#elif ORACLE
          if( (nord->in_nord.s_OL_QUANTITY[nord-
>in_nord.s_O_OL_CNT] = atoi(commandBlock-
>get_ITEM_QTY(itemIndex))) == 0)
#elif SYBASE
          if( (nord->in_nord.s_OL_QUANTITY[nord-
>in_nord.s_O_OL_CNT] = atoi(commandBlock-
>get_ITEM_QTY(itemIndex))) == 0)
#endif
          {

   doNewOrderErrorPage(html,ERR_INVALID_ITEM_OTY,co
mmandBlock,txnHandle);
            return OK;
          }
          else
            itemComplete++;
        }
        //missing previous value of item number
        else if (itemComplete ==1)
        {

   doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,co
mmandBlock,txnHandle);
          return OK;
        }
        //missing 1st value of supp warehouse
        else
        {

   doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_I
D,commandBlock,txnHandle);
          return OK;
        }
      }
      else if(itemComplete==2)     //nothing in the command
block, check to see if the previous item values are present
      {

   doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,co
mmandBlock,txnHandle);
          return OK;
      }

      DEBUGMSG("nord item:" << nord-
>in_nord.s_O_OL_CNT << "SUPPLY_W_ID:" << nord-
>in_nord.s_OL_SUPPLY_W_ID[nord->in_nord.s_O_OL_CNT]
<<
          " OL_I_ID:" << nord->in_nord.s_OL_I_ID[nord-
>in_nord.s_O_OL_CNT] << " OL_QUANTITY:" << nord-
>in_nord.s_OL_QUANTITY[nord->in_nord.s_O_OL_CNT]
<<endl);

      if(itemComplete == 3)
        nord->in_nord.s_O_OL_CNT++;

      itemComplete=0;
    }

    DEBUGMSG("complete nord items:"<<nord-
>in_nord.s_O_OL_CNT<<" initializing remaing unused items
" << NORD_ITEMS - nord->in_nord.s_O_OL_CNT  << " to 0"
<<endl);
    for(int itemIndex=nord-
>in_nord.s_O_OL_CNT;itemIndex<NORD_ITEMS;itemIndex
++)
    {
#ifdef DB2
      nord-
>in_nord.in_item[itemIndex].s_OL_SUPPLY_W_ID=0;
      nord->in_nord.in_item[itemIndex].s_OL_I_ID = 0;
      nord->in_nord.in_item[itemIndex].s_OL_QUANTITY =0;
#elif ORACLE
      nord->in_nord.s_OL_SUPPLY_W_ID[itemIndex]=0;
      nord->in_nord.s_OL_I_ID[itemIndex] = 0;
      nord->in_nord.s_OL_QUANTITY[itemIndex] =0;
#elif SYBASE
      nord->in_nord.s_OL_SUPPLY_W_ID[itemIndex]=0;
      nord->in_nord.s_OL_I_ID[itemIndex] = 0;
      nord->in_nord.s_OL_QUANTITY[itemIndex] =0;
#endif
    }

    DEBUGMSG("nord creating new order results html title
page" <<endl);

    appendText(&html,"<HTML><HEAD><TITLE>TPC-C New
Order Results</TITLE></HEAD>\r\n"
               "<BODY><FORM ACTION=\""
               APP_NAME
               "\" METHOD=\"GET\">\r\n");
    //append menu buttons
    html+=appendButtons(html);
    html+=appendHiddenFields(html,txnHandle);

    appendText(&html,"</FORM><CENTER><H3>New
Order</H3> <BR></CENTER>"
               "<PRE>"
//             "       1       2       3       4       5       6       7
8       9\r\n"
//
"12345678901234567890123456789012345678901234567890123456789012345678
9012345678901234567890123456789012345678901234567890\r\n
               "");
    //assume failure
    nord->out_nord.s_transtatus = 0;

    DEBUGMSG("nord executing COM interface
function,"<<endl<<
      "nord c_id: " <<  nord->in_nord.s_C_ID << endl <<
      "nord w_id: " <<  nord->in_nord.s_W_ID << endl <<
      "nord d_id: " <<  nord->in_nord.s_D_ID << endl);
    HRESULT   hres=0;
    if (txnHandle->comInterface.size > maxDataSize)
    {
      ERRORMSG("[NO]txnHandle->comInterface.size
"<<txnHandle->comInterface.size);
    }
    try
    {
      hres = txnHandle->comInterface.comHandle-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);
    }
    catch(...)
    {
      html+=sprintf(html,"ERROR : nord com call caused
exeception to occur.</PRE></BODY></HTML>");
      ERRORMSG("COM+ exception [NO]txnHandle-
>comInterface.size "<<txnHandle->comInterface.size);
      return OK;
    }

    if(FAILED(hres))
    {
      ERRORMSG("ERROR : nord com call failed, rc:" << hex
<< hres);
      DEBUGMSG("ERROR : nord com call failed, rc:" << hex
<< hres);
    }
```

```
   else
   {
   DEBUGMSG("nord executed OK,"<<endl<<
      "nord c_id: " <<  nord->in_nord.s_C_ID << endl <<
      "nord w_id: " <<  nord->in_nord.s_W_ID << endl <<
      "nord d_id: " <<  nord->in_nord.s_D_ID << endl);
      //com call successful, return object back to pool.
      hres = txnHandle->comInterface.comHandle-
>doSetComplete();
      if(FAILED(hres))
      {
         ERRORMSG("ERROR : nord setcomplete call failed,
rc:" << hex  << hres);
         DEBUGMSG("ERROR : nord setcomplete call failed,
rc:" << hex  << hres);
         return OK;
      }
   }

  nord = (nord_wrapper *)txnHandle->comInterface.txnBuffer;
   DEBUGMSG("nord COM interface function successful,
s_transtatus:" << nord->out_nord.s_transtatus << endl);

   int rc = nord->out_nord.s_transtatus;

   char buffer[10];
   appendText(&html,"Warehouse:  ");
   appendText(&html,itoa(nord-
>in_nord.s_W_ID,buffer,10),6,1);

   appendText(&html,"District: ");
   appendText(&html,itoa(nord-
>in_nord.s_D_ID,buffer,10),26,1);

   appendText(&html,"Date: ");
#ifdef ORACLE
     appendText(&html,nord->out_nord.s_O_ENTRY_D_time);
#elif DB2
      char dateTimeBuffer[50];
      copyOutDateTime(dateTimeBuffer,nord-
>out_nord.s_O_ENTRY_D_time);
      appendText(&html,dateTimeBuffer);
#elif SYBASE
     appendText(&html,nord->out_nord.s_O_ENTRY_D_time);
      DEBUGMSG("Data:" <<nord-
>out_nord.s_O_ENTRY_D_time<< endl);
#endif
   appendText(&html," <BR>"
                "Customer: ");
   appendText(&html,itoa(nord-
>in_nord.s_C_ID,buffer,10),8,1);
   appendText(&html,"Name: ");
   appendText(&html,nord-
>out_nord.s_C_LAST,LAST_NAME_LEN+3,1);

   appendText(&html,"Credit: ");
   appendText(&html,nord->out_nord.s_C_CREDIT,5,1);

   appendText(&html,"%Disc.: ");
   if(rc == OK)
   {
#ifdef ORACLE
      html+=sprintf(html,"%2.2lf",nord-
>out_nord.s_C_DISCOUNT/100.0);
#elif DB2
      html+=sprintf(html,"%2.2lf",nord-
>out_nord.s_C_DISCOUNT/100.0);
#elif SYBASE
      html+=sprintf(html,"%2.2lf",nord-
>out_nord.s_C_DISCOUNT);
#endif
   }
   appendText(&html," <BR>"
            "Order Number: ");
   if(rc != INVALID_STATUS)
      appendText(&html,itoa(nord-
>out_nord.s_O_ID,buffer,10),10,1);

   appendText(&html,"Number of Lines: ");

   if(rc != INVALID_STATUS)
      appendText(&html,itoa(nord-
>out_nord.s_O_OL_CNT,buffer,10),10,1);

   appendText(&html,"W_Tax: ");
   if(rc == OK)
   {
#ifdef ORACLE
      html+=sprintf(html,"%5.2lf",nord-
>out_nord.s_W_TAX/100.0);
#elif DB2
      html+=sprintf(html,"%5.2lf",nord-
>out_nord.s_W_TAX/100.0);
#elif SYBASE
      html+=sprintf(html,"%5.2lf",nord->out_nord.s_W_TAX);
#endif
   }

   appendText(&html,"   D_Tax: ");
   if(rc == OK)
   {
#ifdef ORACLE
      html+=sprintf(html,"%5.2lf",nord-
>out_nord.s_D_TAX/100.0);
#elif DB2
      html+=sprintf(html,"%5.2lf",nord-
>out_nord.s_D_TAX/100.0);
#elif SYBASE
      html+=sprintf(html,"%5.2lf",nord->out_nord.s_D_TAX);
#endif
   }
   appendText(&html," <BR> <BR>"
//          "       1       2       3       4       5       6       7
8       9\r\n"
//
"12345678901234567890123456789012345678901234567890123456789012345678901234567890\r\n"
           " Supp_W  Item_Id   Item_Name                Qty Stock
B/G  Price    Amount <BR> ");

   //display items
   if (rc == OK)
   {
      //display valid items
      for(int itemCount=0;itemCount < nord-
>out_nord.s_O_OL_CNT;itemCount++)
      {
#ifdef DB2
         appendText(&html,itoa(nord-
>in_nord.in_item[itemCount].s_OL_SUPPLY_W_ID,buffer,10
),8,1);
         appendText(&html,itoa(nord-
>in_nord.in_item[itemCount].s_OL_I_ID,buffer,10),10,1);
         appendText(&html,nord-
>out_nord.item[itemCount].s_I_NAME,DEFAULT_STRING_L
EN+1,1);
         appendText(&html,itoa(nord-
>in_nord.in_item[itemCount].s_OL_QUANTITY,buffer,10),5,1
);
         appendText(&html,itoa(nord-
>out_nord.item[itemCount].s_S_QUANTITY,buffer,10),7,1);
         html+=sprintf(html,"%c   $%-7.2lf $%-7.2lf <BR> ",nord-
>out_nord.item[itemCount].s_brand_generic,
                                    nord-
>out_nord.item[itemCount].s_I_PRICE/100.0,
                                    nord-
>out_nord.item[itemCount].s_OL_AMOUNT/100.0);
#elif ORACLE
         appendText(&html,itoa(nord-
>in_nord.s_OL_SUPPLY_W_ID[itemCount],buffer,10),8,1);
         appendText(&html,itoa(nord-
>in_nord.s_OL_I_ID[itemCount],buffer,10),10,1);
```

```
        appendText(&html,nord-
>out_nord.s_I_NAME[itemCount],DEFAULT_STRING_LEN+
1,1);
        appendText(&html,itoa(nord-
>in_nord.s_OL_QUANTITY[itemCount],buffer,10),5,1);
        appendText(&html,itoa(nord-
>out_nord.s_S_QUANTITY[itemCount],buffer,10),7,1);
        html+=sprintf(html,"%c   $%-7.2lf $%-7.2lf <BR> ",nord-
>out_nord.s_brand_generic[itemCount],
                                nord-
>out_nord.s_I_PRICE[itemCount]/100.0,
                                nord-
>out_nord.s_OL_AMOUNT[itemCount]/100.0);
#elif SYBASE
        appendText(&html,itoa(nord-
>in_nord.s_OL_SUPPLY_W_ID[itemCount],buffer,10),8,1);
        appendText(&html,itoa(nord-
>in_nord.s_OL_I_ID[itemCount],buffer,10),10,1);
        appendText(&html,nord-
>out_nord.s_I_NAME[itemCount],DEFAULT_STRING_LEN+
1,1);
        appendText(&html,itoa(nord-
>in_nord.s_OL_QUANTITY[itemCount],buffer,10),5,1);
        appendText(&html,itoa(nord-
>out_nord.s_S_QUANTITY[itemCount],buffer,10),7,1);
        html+=sprintf(html,"%c   $%-7.2lf $%-7.2lf <BR> ",
nord->out_nord.s_brand_generic[itemCount][0],
                                nord-
>out_nord.s_I_PRICE[itemCount],
                                nord-
>out_nord.s_OL_AMOUNT[itemCount]);
#endif
      }
      //display blank line for remaining empty items in the order
      for(int lineBreaks=0;lineBreaks < (NORD_ITEMS-nord-
>out_nord.s_O_OL_CNT);lineBreaks++)
        appendText(&html," <BR>");
   }
   else
      appendText(&html," <BR> <BR> <BR> <BR> <BR>
<BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR> <BR>
<BR>");

   appendText(&html,"\r\n <BR> ");

   html+=displayStatus(html,rc);
   if(rc == OK)
      html+=sprintf(html,"      Total: $%.2lf",nord-
>out_nord.s_total_amount/100.0);
   else
      appendText(&html,"      Total: <BR>");

   appendText(&html,"</PRE></BODY> </HTML>");

   DEBUGMSG("nord html page complete.  returning to
calling function" << endl);

   return OK;
}

/*
**********************************************************************
** Name         : doNewOrderErrorPage
** Description :
**           HTML neworder page entry point
** Parameters:
**           char *     html result page
**           char *     error message
** Returns      :
**           int - return code
** Comments     :
**
**********************************************************************
*/


int doNewOrderErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle)
{
   char *html=htmlPage;

   appendText(&html,"<HTML><HEAD><TITLE>TPC-C New
Order</TITLE></HEAD>\r\n"
              "<BODY><FORM ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">\r\n"
              "<CENTER><H3>Please Fill In New Order
Form.</H3></CENTER>\r\n"
              "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
              CMD_TXN_ID
              "\" VALUE=\""
              CMD_NORD
              "\">");

   //append the hidden warehouse and district fields
   html+=appendHiddenFields(html,txnHandle);

   //int buffer for warehouse
   char buffer[15];
   /*appendText(&html,"<PRE>        1        2        3        4
5        6        7        8        9\r\n"


"12345678901234567890123456789012345678901234567890123456789012345678
90123456789012345678901234567890\r\n"
              "Warehouse: ");*/
   appendText(&html,"<PRE>Warehouse: ");
   appendText(&html,itoa(txnHandle->w_id,buffer,10),7,1);
   appendText(&html,"District: <INPUT NAME=\""
              CMD_D_ID
              "\" SIZE=1>                    Date:<BR>"
                 "Customer <INPUT NAME=\""
              CMD_C_ID
              "\" SIZE=6>  Name:                 Credit:
%Disc.:<BR>"
              "Order Number:          Number of Lines:
W_tax:        D_tax:<BR> <BR>"
              //"       1        2        3        4        5        6        7
8        9\r\n"

//"12345678901234567890123456789012345678901234567
890123456789012345678901234567890\r\n"
              " Supp_W  Item_Num  Item_Name               Qty
Stock  B/G  Price    Amount <BR> ");

   //append the 15 items commands
   html+=appendItems(html,NORD_ITEMS,ITEM_START);
   appendText(&html,message);

   //seal up html page
   appendText(&html,"</PRE></BODY></HTML>");

   return OK;
}


/*
**********************************************************************
** Name         : doPaymentForm
** Description :
**           HTML payment page entry point
** Parameters:
**           htmlPhraser command block
**           char *     html result page
** Returns      :
**           int - return code
** Comments     :
**
**********************************************************************
*/

int doPaymentForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
```

```
   char *html=txnHandle->htmlPage;
   appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Payment</TITLE></HEAD>\r\n"
              "<BODY><FORM ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">\r\n"
              "<CENTER><H3>Please Fill In Payment
Form.</H3></CENTER> <BR>\r\n"
              "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
              CMD_TXN_ID
              "\" VALUE=\""
              CMD_PYMT
              "\">");
   html+=appendHiddenFields(html,txnHandle);
   appendText(&html,"<BR><PRE>\r\n"
              "Date:<BR>"
              "Warehouse: ");
   char buffer[15];
   appendText(&html,itoa(txnHandle->w_id,buffer,10));

   appendSpaces(&html,10);
   appendText(&html,"District: <INPUT NAME=\""
              CMD_D_ID
              "\" SIZE=1>\r\n<BR>"
              "<BR> <BR> <BR>"
              "Customer: "
              "<INPUT NAME=\""
              CMD_C_ID
              "\" SIZE=5>"
              "   "
              "Cust-Warehouse: "
              "<INPUT NAME=\""
              CMD_C_W_ID
              "\" SIZE=5>"
              "   "
              "Cust-District: "
              "<INPUT NAME=\""
              CMD_C_D_ID
              "\" SIZE=1><BR>"
              "Name:  <INPUT NAME=\""
              CMD_C_NAME
              "\" SIZE=20>");
   appendText(&html,"                          Since: <BR>"
              "                            "
              "                         Credit: <BR>"
              "                            "
              "                         %Disc: <BR>"
              "Amount Paid:             "
              "<INPUT NAME=\""
              CMD_AMT_PAID
              "\" SIZE=10>"
              "            "
              "New Cust-Balance:<BR>"
              "Credit Limit:<BR> <BR>Cust-Data:<BR> <BR>
<BR> <BR> </PRE>");

   return OK;
}

/*
*********************************************************************
** Name         : doPaymentResults
** Description :
**           HTML neworder page entry point
** Parameters:
**           htmlPhraser command block
**           char *     html result page
** Returns      :
**            int - return code
** Comments     :
**
*********************************************************************
*/

int doPaymentResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
   char *html=txnHandle->htmlPage;
   char buffer[50];

   struct paym_wrapper  *pymt = NULL;
   pymt = (paym_wrapper*)txnHandle-
>comInterface.txnBuffer;
   ZeroMemory(pymt,sizeof(paym_wrapper));

   //set login warehouse id from command block
   pymt->in_paym.s_W_ID = txnHandle->w_id;

   //set district from command block
#ifdef SYBASE
   if( (pymt->in_paym.s_D_ID =
(CS_TINYINT)atoi(commandBlock->get_D_ID())) == 0)
   {

   doPaymentErrorPage(html,ERR_INVALID_D_ID,command
Block,txnHandle);
      return OK;
   }
#else
   if( (pymt->in_paym.s_D_ID = atoi(commandBlock-
>get_D_ID())) == 0)
   {

   doPaymentErrorPage(html,ERR_INVALID_D_ID,command
Block,txnHandle);
      return OK;
   }
#endif
   //set customer id from command block
   if( (pymt->in_paym.s_C_ID = atoi(commandBlock-
>get_C_ID())) == 0)
   {
      if(*(commandBlock->get_C_NAME()) == NULL)
      {
         //no customer id nor customer last name specified.

   doPaymentErrorPage(html,ERR_MISSING_C_ID_OR_CLA
ST,commandBlock,txnHandle);
         return OK;
      }
      else
         strcpy(pymt->in_paym.s_C_LAST,commandBlock-
>get_C_NAME());
   }
   else
   {
      //make sure that the user only inserted just c_id
      if(*(commandBlock->get_C_NAME()) != NULL)
      {

   doPaymentErrorPage(html,ERR_C_ID_OR_CLAST_ONLY
,commandBlock,txnHandle);
         return OK;
      }
   }

   //get customer warehose id field
   if( (pymt->in_paym.s_C_W_ID = atoi(commandBlock-
>get_C_W_ID())) == 0)
   {

   doPaymentErrorPage(html,ERR_INVALID_C_W_ID,comm
andBlock,txnHandle);
      return OK;
   }

   //get customer district id field
#ifdef SYBASE
   if ( (pymt->in_paym.s_C_D_ID =
(CS_TINYINT)atoi(commandBlock->get_C_D_ID())) == 0)
   {

   doPaymentErrorPage(html,ERR_INVALID_C_D_ID,comma
ndBlock,txnHandle);
```

```
    return OK;
  }
#else
  if ( (pymt->in_paym.s_C_D_ID = atoi(commandBlock-
>get_C_D_ID())) == 0)
  {

  doPaymentErrorPage(html,ERR_INVALID_C_D_ID,comma
ndBlock,txnHandle);
    return OK;
  }
#endif
  if(!copyInMoney32(commandBlock-
>get_AMT_PAID(),&pymt->in_paym.s_H_AMOUNT))
  {

  doPaymentErrorPage(html,ERR_INVALID_PAYMENT_AM
OUNT,commandBlock,txnHandle);
    return OK;
  }

  appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Payment Results</TITLE></HEAD>\r\n"
            "<BODY><FORM ACTION=\""
            APP_NAME
            "\" METHOD=\"GET\">\r\n");
  html+=appendButtons(html);

  html+=appendHiddenFields(html,txnHandle);

  appendText(&html,"</FORM><CENTER><H3>Payment</
H3></CENTER>");

  DEBUGMSG("pymt executing COM interface
function,"<<endl<<
    "pymt c_id: " <<  pymt->in_paym.s_C_ID << endl <<
    "pymt w_id: " <<  pymt->in_paym.s_W_ID << endl <<
    "pymt d_id: " <<  pymt->in_paym.s_D_ID << endl);
  //assume failure
  pymt->out_paym.s_transtatus = OK;

  DEBUGMSG("paym in,"<<endl<<
    "s_W_ID "<<pymt->in_paym.s_W_ID<<endl<<
    "s_C_W_ID "<<pymt->in_paym.s_C_W_ID<<endl<<
    "s_H_AMOUNT "<<pymt-
>in_paym.s_H_AMOUNT<<endl<<
    "s_D_ID "<<(int)pymt->in_paym.s_D_ID<<endl<<
    "s_C_D_ID "<<(int)pymt->in_paym.s_C_D_ID<<endl<<
    "s_C_ID "<<pymt->in_paym.s_C_ID<<endl<<
    "s_W_ID "<<pymt->in_paym.s_W_ID<<endl<<
    "s_C_LAST "<<pymt->in_paym.s_C_LAST<<endl);

  HRESULT  hres=0;
  if (txnHandle->comInterface.size > maxDataSize)
  {
    ERRORMSG("[PY]txnHandle->comInterface.size
"<<txnHandle->comInterface.size);
  }
  try
  {
    hres = txnHandle->comInterface.comHandle-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);
  }
  catch(...)
  {
    html+=sprintf(html,"ERROR : Com Stock call caused
exeception to occur.</PRE></BODY></HTML>");
    ERRORMSG("COM+ exception [PY]txnHandle-
>comInterface.size "<<txnHandle->comInterface.size);
    return OK;
  }

  if(FAILED(hres))
  {
    html+=sprintf(html,"ERROR : pymt com call failed,
rc:%x</PRE></BODY></HTML>",hres);
    ERRORMSG("ERROR : pymt com call failed,
rc:"<<hres<<endl);
    DEBUGMSG("ERROR : pymt com call failed,
rc:"<<hres<<endl);
    return OK;
  }

  DEBUGMSG("pymt executed OK,"<<endl<<
    "pymt c_id: " <<  pymt->in_paym.s_C_ID << endl <<
    "pymt w_id: " <<  pymt->in_paym.s_W_ID << endl <<
    "pymt d_id: " <<  pymt->in_paym.s_D_ID << endl);
  txnHandle->comInterface.comHandle->doSetComplete();
  pymt = (paym_wrapper *)txnHandle-
>comInterface.txnBuffer;

  //get return code
  int rc = pymt->out_paym.s_transtatus;
  if( rc != OK)
  {
    html+=displayStatus(html,rc);
    appendText(&html,"</PRE></BODY></HTML>");
    return OK;
  }
// appendText(&html, "<BR><PRE>\r\n");
// appendText(&html,"       1       2       3       4       5
6       7       8<BR>");

//
  appendText(&html,"12345678901234567890123456789012
34567890123456789012345678901234567890123456789012345678901
<BR>");

  //start creating result body
  appendText(&html, "<BR><PRE>\r\n"
            "Date: ");

#ifdef ORACLE
  appendText(&html,pymt->out_paym.s_H_DATE_time);
#elif DB2
  copyOutDateTime(buffer,pymt-
>out_paym.s_H_DATE_time);
  appendText(&html,buffer);
#elif SYBASE
  appendText(&html,pymt->out_paym.s_H_DATE_time);
#endif
  appendText(&html, "<BR>"
              "Warehouse: ");
  appendText(&html,itoa(pymt-
>in_paym.s_W_ID,buffer,10),6+24,1);
  appendText(&html,"District: ");
  appendText(&html,itoa(pymt-
>in_paym.s_D_ID,buffer,10),2,1);
  appendText(&html,"<BR>");

  //print out warehouse and district information
  appendText(&html,pymt-
>out_paym.s_W_STREET_1,STREET_LEN+21,1);
  appendText(&html,pymt-
>out_paym.s_D_STREET_1,STREET_LEN,1);
  appendText(&html,"<BR>");

  appendText(&html,pymt-
>out_paym.s_W_STREET_2,STREET_LEN+21,1);
  appendText(&html,pymt-
>out_paym.s_D_STREET_2,STREET_LEN,1);
  appendText(&html,"<BR>");

  appendText(&html,pymt-
>out_paym.s_W_CITY,CITY_LEN+1,1);
  appendText(&html,pymt-
>out_paym.s_W_STATE,STATE_LEN+1,1);
  copyOutZip(buffer,pymt->out_paym.s_W_ZIP);
  appendText(&html,buffer);

  appendText(&html,pymt-
>out_paym.s_D_CITY,CITY_LEN+1,1);
  appendText(&html,pymt-
>out_paym.s_D_STATE,STATE_LEN+1,1);
  copyOutZip(buffer,pymt->out_paym.s_D_ZIP);
```

```
   appendText(&html,buffer);

   //print out customer information
   appendText(&html,"<BR> <BR>Customer: ");
   appendText(&html,itoa(pymt-
>out_paym.s_C_ID,buffer,10),5+1,1);

   appendText(&html,"Cust-Warehouse: ");
   appendText(&html,itoa(pymt-
>in_paym.s_C_W_ID,buffer,10),6+1,1);

   appendText(&html,"Cust-District: ");
   appendText(&html,itoa(pymt-
>in_paym.s_C_D_ID,buffer,10));

   //add customer information
   appendText(&html,"<BR>Name:   ");
   appendText(&html,pymt-
>out_paym.s_C_FIRST,FIRST_NAME_LEN+1,1);
   appendText(&html,pymt-
>out_paym.s_C_MIDDLE,INITIALS_LEN+1,1);
   DEBUGMSG("Last name:"<<pymt-
>out_paym.s_C_LAST<<endl);
   appendText(&html,pymt-
>out_paym.s_C_LAST,LAST_NAME_LEN+5,1);

   appendText(&html,"Since: ");
#ifdef ORACLE
   appendText(&html,pymt->out_paym.c_since_d);
#elif DB2
   copyOutDateTime(buffer,pymt-
>out_paym.s_C_SINCE_time);
   appendText(&html,buffer);
#elif SYBASE
   appendText(&html,pymt->out_paym.c_since);
#endif
   appendText(&html,"<BR>");
   appendSpaces(&html,8);

   appendText(&html,pymt-
>out_paym.s_C_STREET_1,STREET_LEN+20,1);
   appendText(&html," Credit: ");
   appendText(&html,pymt->out_paym.s_C_CREDIT);

   appendText(&html,"<BR>");
   appendSpaces(&html,8);

   appendText(&html,pymt-
>out_paym.s_C_STREET_2,STREET_LEN+21,1);
   appendText(&html,"%Disc: ");
   html+=sprintf(html,"%2.2lf",pymt-
>out_paym.s_C_DISCOUNT/100.0);

   appendText(&html,"<BR>");
   appendSpaces(&html,8);

   appendText(&html,pymt-
>out_paym.s_C_CITY,CITY_LEN+1,1);

   appendText(&html,pymt-
>out_paym.s_C_STATE,STATE_LEN+1,1);

   copyOutZip(buffer,pymt->out_paym.s_C_ZIP);
   appendText(&html,buffer,15,1);

   appendText(&html,"Phone: ");
   copyOutPhone(buffer,pymt->out_paym.s_C_PHONE);
   appendText(&html,buffer);

   appendText(&html," <BR> <BR>Amount Paid: $");
   html+=sprintf(html,"%-9.2lf",pymt-
>in_paym.s_H_AMOUNT/100.0);

   appendText(&html," New Cust-Balance: $");
   html+=sprintf(html,"%-9.2lf",pymt-
>out_paym.s_C_BALANCE/100.0);

   appendText(&html,"<BR>Credit Limit: $");
   html+=sprintf(html,"%-9.2lf",pymt-
>out_paym.s_C_CREDIT_LIM/100.0);

   appendText(&html," <BR> <BR>Cust-Data: ");
   if(pymt->out_paym.s_C_CREDIT[0] == 'B' && pymt-
>out_paym.s_C_CREDIT[1] == 'C')
   {
      appendCustData(&html,pymt->out_paym.s_C_DATA);
      appendText(&html," <BR>");
   }
   else
      appendText(&html," <BR> <BR> <BR>");

   html+=displayStatus(html,rc);
   appendText(&html,"</PRE></BODY></HTML>");

   return OK;
}
/*
*****************************************************************
** Name         : doPaymentErrorPage
** Description :
**           append payment error body
** Parameters:
**           char *    html page result
**           char *    error message
**           htmlPhraser * command block
** Returns      :
**           int - return code
** Comments     :
**
*****************************************************************
*/

int doPaymentErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle)
{
   char *html=htmlPage;
   appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Payment</TITLE></HEAD>\r\n"
              "<BODY><FORM ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">\r\n"
              "<CENTER><H3>Please Fill In Payment
Form.</H3></CENTER> <BR>\r\n"
              "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
              CMD_TXN_ID
              "\" VALUE=\""
              CMD_PYMT
              "\">");
   html+=appendHiddenFields(html,txnHandle);
   appendText(&html,"<BR><PRE>\r\n"
              "Date:<BR>"
              "Warehouse: ");
   char buffer[15];
   appendText(&html,itoa(txnHandle->w_id,buffer,10));

   appendSpaces(&html,10);
   appendText(&html,"District: <INPUT NAME=\""
              CMD_D_ID
              "\" SIZE=1>\r\n<BR>"
              "<BR> <BR> <BR> <BR>"
              "Customer: "
              "<INPUT NAME=\""
              CMD_C_ID
              "\" SIZE=5>"
              "   "
              "Cust-Warehouse: "
              "<INPUT NAME=\""
              CMD_C_W_ID
              "\" SIZE=6>"
              "   "
              "Cust-District: "
              "<INPUT NAME=\""
              CMD_C_D_ID
              "\" SIZE=1><BR>"
```

```
            "Name:  <INPUT NAME=\""
            CMD_C_NAME
            "\" SIZE=20>");
  appendText(&html,"                          Since: <BR>"
            "                                "
            "                          Credit: <BR>"
            "                            "
            "                          %Disc: <BR>"
            "Amount Paid:              "
            "<INPUT NAME=\""
            CMD_AMT_PAID
            "\" SIZE=10>"
            "            "
            "New Cust-Balance:<BR>"
            "Credit Limit:<BR> <BR> <BR> Cust-Data:<BR>
<BR> <BR> <BR> ");
  appendText(&html,message);
  appendText(&html,"</PRE>");

  return OK;
}

/*
*********************************************************************
** Name         : doOrderStatusForm
** Description :
**           HTML orderStatus page entry point
** Parameters:
**           htmlPhraser command block
**           char *     html result page
** Returns      :
**           int - return code
** Comments     :
**
*********************************************************************
*/

int doOrderStatusForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
  char *html=txnHandle->htmlPage;

  appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Order Status</TITLE></HEAD>\r\n"
            "<BODY><FORM ACTION=\""
            APP_NAME
            "\" METHOD=\"GET\">\r\n"
            "<CENTER><H3>Please Fill In Order Status
Form.</H3></CENTER> <BR>\r\n"
            "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
            CMD_TXN_ID
            "\" VALUE=\""
            CMD_ORDS
            "\">"
            "<BR> ");
  html+=appendHiddenFields(html,txnHandle);

  appendText(&html,"<PRE>\r\n"
            "Warehouse: ");
  char buffer[15];
  appendText(&html,itoa(txnHandle->w_id,buffer,10));

  appendText(&html,"              District: <INPUT NAME=\""
            CMD_D_ID
            "\" SIZE=1>\r\n<BR>"
            "Customer: "
            "<INPUT NAME=\""
            CMD_C_ID
            "\" SIZE=5>"
            "   "
            "Name: "
            "<INPUT NAME=\""
            CMD_C_NAME
            "\" SIZE=20><BR>"
            "Cust-Balance: <BR>"
            "Order-Number:         Entry-Date:
Carrier-Number<BR>"
            "Supply-W     Item-Num     Oty          Amount
Delivery<BR></PRE>");

  appendText(&html,"</BODY></HTML>");

  return OK;
}

/*
*********************************************************************
** Name         : doOrderStatusResults
** Description :
**           HTML orderStatus page entry point
** Parameters:
**           htmlPhraser*   command block
**           char *     html result page
** Returns      :
**           int - return code
** Comments     :
**
*********************************************************************
*/

int doOrderStatusResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
  char *html=txnHandle->htmlPage;
  struct ords_wrapper *ords = NULL;
 ords = (ords_wrapper *) txnHandle->comInterface.txnBuffer;
  ZeroMemory(ords,sizeof(ords_wrapper));

  //set warehouse login id from command blk
  ords->in_ords.s_W_ID = txnHandle->w_id;

  //set district login id from command blk
  if( (ords->in_ords.s_D_ID = atoi(commandBlock-
>get_D_ID())) == 0)
  {

  doOrderStatusErrorPage(html,ERR_INVALID_D_ID,comm
andBlock,txnHandle);
    return OK;
  }

  if( (ords->in_ords.s_C_ID = atoi(commandBlock-
>get_C_ID())) == 0)
  {
    if(*(commandBlock->get_C_NAME()) == NULL)
    {
      //no customer id nor customer last name specified.

  doOrderStatusErrorPage(html,ERR_MISSING_C_ID_OR_
CLAST,commandBlock,txnHandle);
      return OK;
    }
    else
      strcpy(ords->in_ords.s_C_LAST,commandBlock-
>get_C_NAME());
  }
  else
  {
    //make sure that the user only inserted just c_id
    if(*(commandBlock->get_C_NAME()) != NULL)
    {

  doOrderStatusErrorPage(html,ERR_C_ID_OR_CLAST_ON
LY,commandBlock,txnHandle);
      return OK;
    }
  }

  appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Order Status Results</TITLE></HEAD>\r\n"
            "<BODY><FORM ACTION=\""
            APP_NAME
            "\" METHOD=\"GET\">\r\n");
  html+=appendButtons(html);
```

```
   html+=appendHiddenFields(html,txnHandle);

   appendText(&html,"</FORM>");

   ords->out_ords.s_transtatus = OK;

   DEBUGMSG("ords executing COM interface
function,"<<endl<<
      "ords c_id: " <<  ords->in_ords.s_C_ID << endl <<
      "ords w_id: " <<  ords->in_ords.s_W_ID << endl <<
      "ords d_id: " <<  ords->in_ords.s_D_ID << endl);
   HRESULT   hres=0;
   if (txnHandle->comInterface.size > maxDataSize)
   {
      ERRORMSG("[OS]txnHandle->comInterface.size
"<<txnHandle->comInterface.size);
   }
   try
   {
      hres = txnHandle->comInterface.comHandle-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);
   }
   catch(...)
   {
      html+=sprintf(html,"ERROR : ords com call caused
exeception.</PRE></BODY></HTML>");
      ERRORMSG("COM+ exception [OS]txnHandle-
>comInterface.size "<<txnHandle->comInterface.size);
      return OK;
   }

   if(FAILED(hres))
   {
      html+=sprintf(html,"ERROR : ords com call failed,
rc:%x</PRE></BODY></HTML>",hres);
      ERRORMSG("ERROR : ords com call failed,
rc:"<<DEBUGADDRESS(hres));
      DEBUGMSG("ERROR : ords com call failed,
rc:"<<DEBUGADDRESS(hres));
      return OK;
   }
   DEBUGMSG("ords executed OK,"<<endl<<
      "ords c_id: " <<  ords->in_ords.s_C_ID << endl <<
      "ords w_id: " <<  ords->in_ords.s_W_ID << endl <<
      "ords d_id: " <<  ords->in_ords.s_D_ID << endl);

   txnHandle->comInterface.comHandle->doSetComplete();

   ords = (ords_wrapper *)txnHandle->comInterface.txnBuffer;
   int rc = ords->out_ords.s_transtatus;

   if( rc != OK)
   {
      html+=displayStatus(html,rc);
      appendText(&html,"</PRE></BODY></HTML>");
      return OK;
   }

   //start creating result body
   appendText(&html,"</FORM><CENTER><H3>Order-
Status</H3></CENTER>");
   appendText(&html, "<BR><PRE>\r\n");
// appendText(&html,"         1         2         3         4         5
6         7         8<BR>");
//
   appendText(&html,"12345678901234567890123456789012
345678901234567890123456789012345678901234567890
<BR>");
   appendText(&html, "Warehouse: ");
   char buffer[50];

   appendText(&html,itoa(ords-
>in_ords.s_W_ID,buffer,10),6+1,1);
   appendText(&html,"District: ");
   appendText(&html,itoa(ords->in_ords.s_D_ID,buffer,10));
   appendText(&html,"<BR>"
              "Customer: ");
   //get customer id
   appendText(&html,itoa(ords-
>in_ords.s_C_ID,buffer,10),6+1,1);
   appendText(&html,"Name: ");
   //get first, middle, and last from wrapper
   appendText(&html,ords-
>out_ords.s_C_FIRST,FIRST_NAME_LEN+1,1);
   appendText(&html,ords-
>out_ords.s_C_MIDDLE,INITIALS_LEN+1,1);
   appendText(&html,ords-
>out_ords.s_C_LAST,LAST_NAME_LEN+5,1);

   //get customer balance from wrapper
   appendText(&html,"\r\nCust-Balance: $");
#ifdef SYBASE
   html+=sprintf(html,"%.2lf",ords->out_ords.s_C_BALANCE);
#else
   html+=sprintf(html,"%.2lf",ords-
>out_ords.s_C_BALANCE/100.0);
#endif
   //display order number, entry date, and carrier number
   appendText(&html,"<BR> <BR>"
              "Order-Number ");
   appendText(&html,itoa(ords-
>out_ords.s_O_ID,buffer,10),11,1);
   appendText(&html,"Entry-Date:");
#ifdef ORACLE
   appendText(&html,ords->out_ords.s_O_ENTRY_D_time);
#elif DB2
   copyOutDateTime(buffer,ords-
>out_ords.s_O_ENTRY_D_time);
   appendText(&html,buffer,22,1);
#elif SYBASE
   appendText(&html,ords->out_ords.s_O_ENTRY_D_time);
#endif
   appendText(&html,"Carrier-Number: ");
   appendText(&html,itoa(ords-
>out_ords.s_O_CARRIER_ID,buffer,10));

   //add item title columns
   appendText(&html,"<BR>"
              "Supply-W     "
              "Item-Id    "
              "Qty    "
              "Amount      "
              "Delivery-Date<BR>");

   //display items
   for (int itemCount=0;itemCount<ords-
>out_ords.s_ol_cnt;itemCount++)
   {
      appendSpaces(&html,2);

#ifdef DB2
      //get supp w
      appendText(&html,itoa(ords-
>out_ords.item[itemCount].s_OL_SUPPLY_W_ID,buffer,10),
11,1);

      //get item num
      appendText(&html,itoa(ords-
>out_ords.item[itemCount].s_OL_I_ID,buffer,10),11,1);

      //get item oty
      appendText(&html,itoa(ords-
>out_ords.item[itemCount].s_OL_QUANTITY,buffer,10),6,1);

      //get item dollor amount
      html+=sprintf(html,"$%-14.2lf",ords-
>out_ords.item[itemCount].s_OL_AMOUNT/100.0);
#elif ORACLE
      //get supp w
      appendText(&html,itoa(ords-
>out_ords.s_OL_SUPPLY_W_ID[itemCount],buffer,10),11,1);

      //get item num
```

```
    appendText(&html,itoa(ords-
>out_ords.s_OL_I_ID[itemCount],buffer,10),11,1);

    //get item oty
    appendText(&html,itoa(ords-
>out_ords.s_OL_QUANTITY[itemCount],buffer,10),6,1);

    //get item dollor amount
    html+=sprintf(html,"$%-14.2lf",ords-
>out_ords.s_OL_AMOUNT[itemCount]/100.0);
#elif SYBASE
    //get supp w
    appendText(&html,itoa(ords-
>out_ords.s_OL_SUPPLY_W_ID[itemCount],buffer,10),11,1);

    //get item num
    appendText(&html,itoa(ords-
>out_ords.s_OL_I_ID[itemCount],buffer,10),11,1);

    //get item oty
    appendText(&html,itoa(ords-
>out_ords.s_OL_QUANTITY[itemCount],buffer,10),6,1);

    //get item dollor amount
    html+=sprintf(html,"$%-14.2lf",ords-
>out_ords.s_OL_AMOUNT[itemCount]);
#endif

    //get delivery date
#ifdef ORACLE
    appendText(&html,ords-
>out_ords.s_OL_DELIVERY_D_time[itemCount]);
#elif DB2
    copyOutDate(buffer,ords-
>out_ords.item[itemCount].s_OL_DELIVERY_D_time);
    appendText(&html,buffer);
#elif SYBASE
    appendText(&html,ords-
>out_ords.s_OL_DELIVERY_D_time[itemCount]);
#endif
    appendText(&html," <BR>");

  }

  //append line breaks if item count is less than 15
  for (int itemCount=0;itemCount < (15-ords-
>out_ords.s_ol_cnt);itemCount++)
    appendText(&html,"<BR> ");

  html+=displayStatus(html,rc);

  appendText(&html,"</PRE></BODY></HTML>");

  return OK;
}

/*
**********************************************************************
** Name        : doOrderStatusErrorPage
** Description :
**          HTML orderStatus error page
** Parameters:
**          char *    html page result
**          char *    error message
**          htmlPhraser*   command block
**          TXN_HANDLE*  txn handle
** Returns     :
**          int - return code
** Comments    :
**
**********************************************************************
*/

int doOrderStatusErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle)
{
  char *html=htmlPage;

  appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Order Status</TITLE></HEAD>\r\n"
            "<BODY><FORM ACTION=\""
            APP_NAME
            "\" METHOD=\"GET\">\r\n"
            "<CENTER><H3>Please Fill In Order Status
Form.</H3></CENTER> <BR>\r\n"
            "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
            CMD_TXN_ID
            "\" VALUE=\""
            CMD_ORDS
            "\">"
            "<BR> ");
  html+=appendHiddenFields(html,txnHandle);

  appendText(&html,"<PRE>\r\n"
            "Warehouse: ");
  char buffer[15];
  appendText(&html,itoa(txnHandle->w_id,buffer,10));

  appendText(&html,"            District: <INPUT NAME=\""
            CMD_D_ID
            "\" SIZE=1>\r\n<BR>"
            "Customer: "
            "<INPUT NAME=\""
            CMD_C_ID
            "\" SIZE=5>"
            "   "
            "Name: "
            "<INPUT NAME=\""
            CMD_C_NAME
            "\" SIZE=20><BR>"
            "Cust-Balance: <BR>"
            "Order-Number:          Entry-Date:
Carrier-Number<BR>"
            "Supply-W    Item-Num    Oty         Amount
Delivery <BR>");

  appendText(&html,message);
  appendText(&html,"</PRE></BODY></HTML>");

  return OK;
}

/*
**********************************************************************
** Name        : doDeliveryForm
** Description :
**          HTML payment page entry point
** Parameters:
**          htmlPhraser command block
**          char *    html result page
** Returns     :
**          int - return code
** Comments    :
**
**********************************************************************
*/
int doDeliveryForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
  char *html=txnHandle->htmlPage;

  appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Delivery</TITLE></HEAD>\r\n"
            "<BODY><FORM ACTION=\""
            APP_NAME
            "\" METHOD=\"GET\">\r\n"
            "<CENTER><H3>Delivery.</H3></CENTER>\r\n"
            "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
            CMD_TXN_ID
            "\" VALUE=\""
            CMD_DLVY
            "\">");
  html+=appendHiddenFields(html,txnHandle);
```

```
   appendText(&html,"<BR> <PRE>"
              "Warehouse: ");
   char buffer[10];
   appendText(&html,itoa(txnHandle->w_id,buffer,10));

   appendText(&html," <BR> <BR>"
              "Carrier Number: "
              "<INPUT NAME=\""
              CMD_CARRIER_NUM
              "\" SIZE=1>"
              "</FORM></PRE>");

   appendText(&html,"</BODY></HTML>");

   return OK;
}

/*
*********************************************************************
** Name         : doDeliveryResults
** Description :
**           HTML payment page entry point
** Parameters:
**           htmlPhraser*   command block
**           TXN_HANDLE*  txn handle
** Returns      :
**           int - return code
** Comments     :
**
*********************************************************************
*/

int doDeliveryResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
   char *html = txnHandle->htmlPage;

   //declare delivery structure
   struct dlvy_wrapper     dlvy;

   //set warehouse login id from command blk
   dlvy.in_dlvy.s_W_ID = txnHandle->w_id;

   //set the carrier id from command blk
   if( (dlvy.in_dlvy.s_O_CARRIER_ID = atoi(commandBlock-
>get_CARRIER_NUM())) == 0)
   {

   doDeliveryErrorPage(html,ERR_INVALID_CARRIER,comm
andBlock,txnHandle);
      return OK;
   }

   //print title, add hidden fields , txn buttons
   appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Delivery Results</TITLE></HEAD>\r\n<BODY><FORM
ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">\r\n");

   html+=appendButtons(html);

   html+=appendHiddenFields(html,txnHandle);

   appendText(&html,
"<FORM><CENTER><H3>Delivery</H3></CENTER>");

   //call null db or comm w/ delivery wrapper
   int rc =
queueDlvyTxn(dlvy.in_dlvy.s_W_ID,dlvy.in_dlvy.s_O_CARRI
ER_ID);
   //if we are using the null db, rc will always be ok.  The only
time rc != OK is
   //1) unable to queue txn because dlvy queue is full.
   if( rc != OK)
   {
      html+=displayStatus(html,rc);
      appendText(&html,"</PRE></BODY></HTML>\r\n");

      ERRORMSG("ERROR : Unable to queue dlvy txn,
rc:"<<rc<<endl);
      return OK;
   }

   //start creating result body
   appendText(&html,"Warehouse: ");

   //get w_id from wrapper
   char buffer[10];
   appendText(&html,itoa(dlvy.in_dlvy.s_W_ID,buffer,10));
   appendText(&html,"<BR> <BR>Carrier Number: ");

   //get carrier_id from wrapper
   appendText(&html,itoa(dlvy.in_dlvy.s_O_CARRIER_ID,buff
er,10));
   appendText(&html,"<BR> <BR>Execution Status: Delivery
has been queued </PRE></BODY></HTML>");

   return OK;
}

/*
*********************************************************************
** Name         : doDeliveryErrorPage
** Description :
**           HTML payment error page entry point
** Parameters:
**           char *     html result page
**           char *     error message
**           htmlPhraser command block
**           TXN_HANDLE*  txn handle
**
** Returns      :
**           int - return code
** Comments     :
**
*********************************************************************
*/
int doDeliveryErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle)
{
   char *html=htmlPage;

   appendText(&html,"<HTML><HEAD><TITLE>TPC-C
Delivery</TITLE></HEAD>\r\n"
              "<BODY><FORM ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">\r\n"
              "<CENTER><H3>Delivery.</H3></CENTER>\r\n"
              "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
              CMD_TXN_ID
              "\" VALUE=\""
              CMD_DLVY
              "\">");
   html+=appendHiddenFields(html,txnHandle);

   appendText(&html,"<BR> <PRE>"
              "Warehouse: ");
   char buffer[15];
   appendText(&html,itoa(txnHandle->w_id,buffer,10));

   appendText(&html," <BR> <BR>"
              "Carrier Number: "
              "<INPUT NAME=\""
              CMD_CARRIER_NUM
              "\" SIZE=1> <BR>");

   appendText(&html,message);
   appendText(&html,"</PRE></BODY></HTML>");

   return OK;
}
```

```
/*
*********************************************************************
** Name         : doStockForm
** Description :
**           HTML stock page entry point
** Parameters:
**           htmlPhraser command block
**           TXN_HANDLE*  txn handle
** Returns      :
**           int - return code
** Comments     :
**
*********************************************************************
*/

int doStockForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
   char *html=txnHandle->htmlPage;
   appendText(&html,"<HTML><HEAD><TITLE>TPC-C Stock
Level</TITLE></HEAD>\r\n"
              "<BODY><FORM ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">\r\n"
              "<CENTER><H3>Please Fill In Stock
Form.</H3></CENTER> <BR>\r\n"
              "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
              CMD_TXN_ID
              "\" VALUE=\""
              CMD_STOK
              "\">");
   html+=appendHiddenFields(html,txnHandle);

   appendText(&html,"<PRE>"
              "Warehouse: ");
   char buffer[15];
   appendText(&html,itoa(txnHandle->w_id,buffer,10),6+1,1);
   appendText(&html,"District: ");

   appendText(&html,itoa(txnHandle->d_id,buffer,10));
   appendText(&html," <BR> <BR>"
              "Stock Level Threshold: "
              "<INPUT NAME=\""
              CMD_STK_THRESHOLD
              "\" SIZE=1> <BR> <BR>"
              "Low Stock: <BR>"
              "</PRE>");

   appendText(&html,"</FORM></BODY></HTML>");
   return OK;
}

/*
*********************************************************************
** Name         : doStockResults
** Description :
**              HTML stock page entry point
** Parameters:
**           htmlPhraser command block
**           TXN_HANDLE * handle for this transaction
** Returns      :
**           int - return code
** Comments     :
**
*********************************************************************
*/

int doStockResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle)
{
   char *html = txnHandle->htmlPage;

   struct stok_wrapper *stok;
   stok = (stok_wrapper*)txnHandle->comInterface.txnBuffer;
   ZeroMemory(stok,sizeof(stok_wrapper));

   //set warehouse login id from command blk
   stok->in_stok.s_W_ID = txnHandle->w_id;

   //set district login id from command blk
   stok->in_stok.s_D_ID = txnHandle->d_id;

   //set stock level threshold id from command blk
   if( (stok->in_stok.s_threshold = atoi(commandBlock-
>get_STK_THRESHOLD())) == 0)
   {

   doStockErrorPage(html,ERR_INVALID_THRESHOLD,com
mandBlock,txnHandle);
      return OK;
   }
   //assume failure, set s_transtatus to err
   stok->out_stok.s_transtatus = INVALID_STATUS;

   //print title, add hidden fields , txn buttons
   appendText(&html,"<HTML><HEAD><TITLE>TPC-C Stock
Level Results</TITLE></HEAD>\r\n"
              "<BODY><FORM ACTION=\""
              APP_NAME
              "\" METHOD=\"GET\">\r\n");
   html+=appendButtons(html);

   html+=appendHiddenFields(html,txnHandle);

   appendText(&html,"</FORM>");

   stok->out_stok.s_transtatus = OK;

   DEBUGMSG("stok executing COM interface
function,"<<endl<<
      "stok d_id: " <<  stok->in_stok.s_D_ID << endl <<
      "stok w_id: " <<  stok->in_stok.s_W_ID << endl <<
      "stok s_threshold: " <<  stok->in_stok.s_threshold <<
endl);

   HRESULT   hres=0;
   if (txnHandle->comInterface.size > maxDataSize)
   {
      ERRORMSG("[SR]txnHandle->comInterface.size
"<<txnHandle->comInterface.size);
   }
   try
   {
      hres = txnHandle->comInterface.comHandle-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);
   }
   catch(...)
   {
      html+=sprintf(html,"ERROR : Com Stock call caused
exeception to occur.</PRE></BODY></HTML>");
      ERRORMSG("COM+ exception [SR]txnHandle-
>comInterface.size "<<txnHandle->comInterface.size);
      return OK;
   }

   //cast result back to stock structure
   if(FAILED(hres))
   {
      html+=sprintf(html,"ERROR : stok com call failed,
rc:%ld</PRE></BODY></HTML>",hres);
      ERRORMSG("ERROR : stok com call failed,
rc:"<<DEBUGADDRESS(hres)<<endl);
      DEBUGMSG("ERROR : stok com call failed,
rc:"<<DEBUGADDRESS(hres)<<endl);
      return OK;
   }

   DEBUGMSG("stok executed OK,"<<endl<<
      "stok d_id: " <<  stok->in_stok.s_D_ID << endl <<
```

```
    "stok w_id: " <<  stok->in_stok.s_W_ID << endl <<
    "stok s_threshold: " <<  stok->in_stok.s_threshold <<
endl);
  try
  {
    txnHandle->comInterface.comHandle->doSetComplete();
  }
  catch(...)
  {
    ERRORMSG("txnHandle address:"<<hex<<txnHandle<<
      "txnHandle-
>comInterface.comHandle:"<<hex<<txnHandle-
>comInterface.comHandle<<
      "txnHandle->comInterface.txnBuffer:"<<hex<<(void
*)txnHandle->comInterface.txnBuffer<<endl);


    ERRORMSG("Com Stock setComplete call caused
exeception to occur."<<endl);
    html+=sprintf(html,"ERROR : Com Stock setComplete
call caused exeception to occur.</PRE></BODY></HTML>");
    return OK;
  }

  stok = (stok_wrapper *)txnHandle->comInterface.txnBuffer;
  int rc = stok->out_stok.s_transtatus;
  if(rc != OK)
  {
    html+=displayStatus(html,rc);
    appendText(&html,"</PRE></BODY></HTML>");
    return OK;
  }

  //start creating result body
  appendText(&html,"<FORM><CENTER><H3>Stock-
Level</H3></CENTER>");
  appendText(&html, "<BR><PRE>\r\n"
    "Warehouse: ");

  //get w_id from wrapper
  char buffer[10];
  appendText(&html,itoa(stok-
>in_stok.s_W_ID,buffer,10),6+1,1);

  appendText(&html,"District: ");
  appendText(&html,itoa(stok->in_stok.s_D_ID,buffer,10));

  appendText(&html," <BR> <BR>"
            "Stock Level Threshold: ");
  appendText(&html,itoa(stok-
>in_stok.s_threshold,buffer,10));

  appendText(&html," <BR> <BR>"
            "Low Stock: ");
  appendText(&html,itoa(stok-
>out_stok.s_low_stock,buffer,10));
  appendText(&html," <BR> <BR>");

  html+=displayStatus(html,rc);
  appendText(&html,"</PRE></BODY></HTML>");

  return OK;
}

/*
*****************************************************************
** Name         : doStockErrorPage
** Description :
**              HTML stock page entry point
** Parameters:
**           htmlPhraser command block
**           char *     html result page
**             char *     query string
**             tpccHandle *handle for this transaction
** Returns      :
**             int - return code
** Comments     :
**
*****************************************************************
*/

int doStockErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle)
{
  char *html=htmlPage;

  appendText(&html,"<HTML><HEAD><TITLE>TPC-C Stock
Level</TITLE></HEAD>\r\n"
            "<BODY><FORM ACTION=\""
            APP_NAME
            "\" METHOD=\"GET\">\r\n"
            "<CENTER><H3>Please Fill In Stock
Form.</H3></CENTER> <BR>\r\n"
            "Submit Transaction <INPUT TYPE=\"submit\"
NAME=\""
            CMD_TXN_ID
            "\" VALUE=\""
            CMD_STOK
            "\">");
  html+=appendHiddenFields(html,txnHandle);

  appendText(&html,"<PRE>"
            "Warehouse:  ");
  char buffer[15];
  appendText(&html,itoa(txnHandle->w_id,buffer,10));
  appendSpaces(&html,2);
  appendText(&html,"District: ");
  appendText(&html,commandBlock->get_D_ID());
  appendText(&html," <BR> <BR>"
            "Stock Level Threshold: "
            "<INPUT NAME=\""
            CMD_STK_THRESHOLD
            "\" SIZE=1> <BR> <BR>"
            "Low Stock: <BR>");

  appendText(&html,message);


  appendText(&html,"</PRE></FORM></BODY></HTML>");

  return OK;
}



/*
*****************************************************************
** Name         : doExit
** Description :
**           HTML exit page entry point
** Parameters:
**           htmlPhraser command block
**           char *     html result page
** Returns      :
**           int - return code
** Comments     :
**
*****************************************************************
*/
int doExit(htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle)
{
  return (doLoginForm(commandBlock,txnHandle));
}

/*
*****************************************************************
** Name         : displayStatus
** Description :
**           appends status string to the html page
** Parameters:
**           char*      html page
**           int    rc
** Returns      :
**           amount of characters the function appened
```

```
**          to the html page
** Comments    :
**
*****************************************************************
*/
int displayStatus(char *htmlPage,int rc)
{
  char *html = htmlPage;

  appendText(&html,"");

  switch (rc)
  {
  case OK:
    appendText(&html,"Execution Status: Transaction
Committed",50,1);
    break;
  case INVALID_ITEM:
    appendText(&html,"Execution Status: Item number is not
valid",50,1);
    break;
  case INVALID_STATUS:
    appendText(&html,"Execution Status: ERROR Rollback
INVALID_STATUS",50,1);
    break;
  case INVALID_COM_STATUS:
    appendText(&html,"Execution Status: ERROR: Rollback
COM FAILURE",50,1);
    break;
  case ERR_DLVY_QUEUE_FULL:
    appendText(&html,"Execution Status: ERROR: Rollback
DLVY QUEUE FULL",50,1);
    break;
  default:
    appendText(&html,"Execution Status: ERROR:
Rollback",50,1);

  };

  appendText(&html,"    ");

  return (int)(html - htmlPage);
}
/*
*****************************************************************
** Name        : appendButtons
** Description :
**          append hidden field to recognize user after login
**
** Parameters:
**          *htmlPage      html result page
**          *TXN_HANDLE    txn handle
** Returns    :
**          int        amount of characters the function appened
**                     to the html page
** Comments    :
**
*****************************************************************
*/
int appendHiddenFields(char *htmlPage,TXN_HANDLE
*txnHandle)
{
  char *html = htmlPage;
  char buffer[15];

  appendText(&html,"<INPUT TYPE=\"hidden\" NAME=\""
             CMD_TERM_ID
             "\" VALUE=\"");
  appendText(&html,itoa(txnHandle->term_id,buffer,10));
  appendText(&html,"\">\r\n");

  return (int)(html-htmlPage);

}
/*
*****************************************************************
** Name        : appendButtons
** Description :
**          appends buttons transaction buttons to result page
** Parameters:
**          *htmlPage
**
** Returns    :
**          amount of characters the function appened
**          to the html page
** Comments    :
**
*****************************************************************
*/
int appendButtons(char *htmlPage)
{
  char *html = htmlPage;

  appendText(&html,"<INPUT TYPE=\"submit\" NAME=\""
             CMD_TXN_ID
             "\" VALUE=\""
             CMD_NORD
             "\">\r\n"
             "<INPUT TYPE=\"submit\" NAME=\""
             CMD_TXN_ID
             "\" VALUE=\""
             CMD_PYMT
             "\">\r\n"
             "<INPUT TYPE=\"submit\" NAME=\""
             CMD_TXN_ID
             "\" VALUE=\""
             CMD_ORDS
             "\">\r\n"
             "<INPUT TYPE=\"submit\" NAME=\""
             CMD_TXN_ID
             "\" VALUE=\""
             CMD_DLVY
             "\">\r\n"
             "<INPUT TYPE=\"submit\" NAME=\""
             CMD_TXN_ID
             "\" VALUE=\""
             CMD_STOK
             "\">\r\n"
             "<INPUT TYPE=\"submit\" NAME=\""
             CMD_TXN_ID
             "\" VALUE=\""
             CMD_EXIT
            "\">\r\n <BR>");

  return (int)(html - htmlPage);
}

/*
*****************************************************************
** Name        : appendItems
** Description :
**          appends items to new order and order status page
** Parameters:
**          *htmlPage        html result page
**          short            items to append
**          short            item CMD id start
**
** Returns    :
**          amount of characters the function appened
**          to the html page
** Comments    :
**
*****************************************************************
*/
int appendItems(char *htmlPage,short itemCount,short
cmdIDStart)
{
  char *html = htmlPage;
  char numBuffer[MAX_INT_BUFFER];

  for(int item=0;item < itemCount;item++)
  {
    appendText(&html,"<BR> <INPUT NAME=\"");
    appendText(&html,itoa(cmdIDStart++,numBuffer,10));
    appendText(&html,"\" SIZE=6> <INPUT NAME=\"");
    appendText(&html,itoa(cmdIDStart++,numBuffer,10));
```

```
    appendText(&html,"\" SIZE=6>                    <INPUT
NAME=\"");
    appendText(&html,itoa(cmdIDStart++,numBuffer,10));
    appendText(&html,"\" SIZE=2>\r\n");
  }

  return (int)(html - htmlPage);
}

/*
*********************************************************************
** Name        : dlvyThreadEntry
** Description :
**          dlvy thread worker entry point
** Parameters:
**
** Returns     :
**
** Comments    :
**          All dlvy threads created by initDly enter at
**          this point.  They must first make a connection
**          to the database, then go to sleep.
**
**          Main isapi threads control dlvy worker semaphore
**          and signal when a dlvy txn is queued.
**
*********************************************************************
*/

void dlvyThreadEntry(void *)
{
  int     rc = 0,count=0;
   bool bRetry = false;
  DEBUGMSG("dlvyThread " << GetCurrentThreadId() << "
entered dlvyThreadEntry, calling db_connect to db:" <<
dbName << endl);

  void *connectHandle;
  //connect to database.
  do {
    bRetry = false;
    DEBUGMSG("ptr created. calling db_connect to db:" <<
dbName << endl);
    ERRORMSG("ptr created. calling db_connect to db:" <<
dbName << endl);
    rc = db_connect(dbName,&connectHandle);

    if(rc != OK)
    {
      ERRORMSG("dlvyThread " << GetCurrentThreadId()
<<" unable to connect to database, rc:" << rc << endl);
      DEBUGMSG("dlvyThread " << GetCurrentThreadId()
<<" unable to connect to database, rc:" << rc << endl);
      bRetry = true;
      if (count++ > 100)
        return;
      Sleep(50);
    }
  } while (bRetry);
  DEBUGMSG("dlvyThread " << GetCurrentThreadId() << "
connect to db:" << dbName << " successful" << endl);

  FILE *dlvyLog = NULL;
  char logFileName[MAX_STRING_LEN] = {NULL};

  EnterCriticalSection(&isapiLock);
  //open dlvy log file for this thread
  sprintf(logFileName,"%s\\del_%d.txt",dlvyLogPath,dlvyThre
adID);
  dlvyLog = fopen(logFileName,"w");
  if(!dlvyLog)
  {
    ERRORMSG("dlvyThread " << GetCurrentThreadId() <<
" unable to open dlvy log "
          << dlvyLogPath << "\\del_" << dlvyThreadID <<
endl);
    DEBUGMSG("dlvyThread " << GetCurrentThreadId() <<
" unable toopen dlvy log "
          << dlvyLogPath << "\\del_" << dlvyThreadID <<
endl);
    return;
  }

  //increment the global dlvy thread id
  dlvyThreadID++;

  LeaveCriticalSection(&isapiLock);

  DEBUGMSG("dlvyThread " << GetCurrentThreadId() <<"
dlvy log file name: " << logFileName << " open." << endl);

  HANDLE  workerHandles[2];              //handle array to
store event to wait on

  struct  DLVYQUEUEDATA    dlvyQueueData;    //dlvy
queue struct to store queued txn
  struct dlvy_wrapper    dlvyTxn;          //dlvy wrapper of db2
structs

  struct  _timeb        endQueueTime;     //time stamp to
queue removal time
  struct  _timeb        endProcessTime;     //time stamp for
end process time
  char  orderIDs[MAX_STRING_LEN] = {NULL};
  //string to store oids for each district
  int     bytesWritten =  0;
  int     dlvyCount =0;

  DEBUGMSG("dlvyThread entering work loop" << endl);

  //successful, while true
  while(true)
  {
//    try
//    {
      DEBUGMSG("dlvyThread initializing wait handles" <<
endl);

      //wait for both program exit AND if there is work to do
      workerHandles[0] = dlvyThreadDone;
      workerHandles[1] = dlvyThreadSemaphore;

      DEBUGMSG("dlvyThread going to sleep waiting for
wrk" << endl);

      rc =
WaitForMultipleObjects(2,&workerHandles[0],FALSE,INFINIT
E);

      DEBUGMSG("dlvyThread awake, checking wake
condition" << endl);

      if(rc == WAIT_OBJECT_0)
        break;
      else if(rc == (WAIT_OBJECT_0+1) )
      {
        DEBUGMSG("dlvyThread awake, wake condition of
dlvyThreadSemaphore" << endl);
      }

      DEBUGMSG("dlvyThread trying to enter critical
section" << endl);

      EnterCriticalSection(&dlvyQueueLock);

      DEBUGMSG("dlvyThread entered critical section" <<
endl);

      //remove queued dlvy txn
      dlvyQueueData.enqueueTime.time     =
  dlvyQueue[dlvyBufferThreadIndex].enqueueTime.time;
      dlvyQueueData.enqueueTime.millitm  =
  dlvyQueue[dlvyBufferThreadIndex].enqueueTime.millitm;
```

```
        dlvyQueueData.in_s_0_CARRIER_ID    =
   dlvyQueue[dlvyBufferThreadIndex].in_s_0_CARRIER_ID;
        dlvyQueueData.warehouse          =
   dlvyQueue[dlvyBufferThreadIndex].warehouse;

        DEBUGMSG("dlvyThread removed dlvy:" << dlvyCount
<< ",w_id:" << dlvyQueueData.warehouse
                    << " carrier_id:" <<
dlvyQueueData.in_s_0_CARRIER_ID << endl);

        DEBUGMSG("dlvyThread removed dlvy in queue index:
" <<dlvyBufferThreadIndex<< " w_id: " <<
dlvyQueueData.warehouse
                    << " carrier_id: " <<
dlvyQueueData.in_s_0_CARRIER_ID << endl);

        //increment the number of free slots
        dlvyBufferFreeSlots++;

        //increment the thread index to next slot in dlvy queue
        dlvyBufferThreadIndex++;

        DEBUGMSG("dlvyThread incremented amount of free
slots:" << dlvyBufferFreeSlots << " and thread index:" <<
                    dlvyBufferThreadIndex << endl);

        //check if we reached the end of dlvy queue, if so, reset
back index back to 0
        if(dlvyBufferThreadIndex == dlvyQueueLen)
        {
           DEBUGMSG("dlvyThread  reset
dlvyBufferThreadIndex to 0, current dlvyBufferThreadIndex:"
<< dlvyBufferThreadIndex
                << " free slots:"<<dlvyBufferFreeSlots<<endl);
           ERRORMSG("dlvyThread  reset
dlvyBufferThreadIndex to 0, current dlvyBufferThreadIndex:"
<< dlvyBufferThreadIndex
                << " free slots:"<<dlvyBufferFreeSlots<<endl);

           dlvyBufferThreadIndex=0;
        }

        DEBUGMSG("dlvyThread releasing critical section" <<
endl);

        LeaveCriticalSection(&dlvyQueueLock);

        //take enqueue time
        _ftime(&endQueueTime);

        DEBUGMSG("dlvyThread executing txn w_id:" <<
dlvyQueueData.warehouse
                << " carrier_id:" <<
dlvyQueueData.in_s_0_CARRIER_ID << endl);

        //prepare to call database
        dlvyTxn.in_dlvy.s_O_CARRIER_ID  =
   dlvyQueueData.in_s_0_CARRIER_ID;
        dlvyTxn.in_dlvy.s_W_ID       =
   dlvyQueueData.warehouse;
#ifdef SYBASE
        dlvyTxn.in_dlvy.s_D_ID       = 1;
#endif
        dlvyTxn.out_dlvy.s_transtatus =  OK;

        //increment dlvy count
        dlvyCount++;

        DEBUGMSG("dlvyThread %d calling dlvy txn" << rc <<
endl);

        //call dvly txn
        rc = dlvyCall(&dlvyTxn,connectHandle);

        _ftime(&endProcessTime);

        rc = dlvyTxn.out_dlvy.s_transtatus;

#ifdef ORACLE
        DEBUGMSG("dlvy txn response time:"<<
                (((endProcessTime.time -
endQueueTime.time)*1000)+
                 (endProcessTime.millitm -
endQueueTime.millitm))/1000.0)<<
                " w_id:"<<dlvyTxn.in_dlvy.s_W_ID<<" carrier:"
<<dlvyTxn.in_dlvy.s_O_CARRIER_ID<<
                " rc: "<< rc <<endl);
#elif DB2
        DEBUGMSG("dlvy txn response time:"<<
                (((endProcessTime.time -
endQueueTime.time)*1000)+
                 (endProcessTime.millitm -
endQueueTime.millitm))/1000.0)<<
                " w_id:"<<dlvyTxn.in_dlvy.s_W_ID<<" carrier:"
<<dlvyTxn.in_dlvy.s_O_CARRIER_ID<<
                " deadLocks:"<<dlvyTxn.out_dlvy.deadlocks<<" rc:
"<< rc <<endl);
#elif SYBASE
        DEBUGMSG("dlvy txn response time:"<<
                (((endProcessTime.time -
endQueueTime.time)*1000)+
                 (endProcessTime.millitm -
endQueueTime.millitm))/1000.0)<<
                " w_id:"<<dlvyTxn.in_dlvy.s_W_ID<<" carrier:"
<<dlvyTxn.in_dlvy.s_O_CARRIER_ID<<
                " rc: "<< rc <<endl);
#endif
        DEBUGMSG("dlvyThread dlvy s_transtatus:" << rc <<
endl);

        if(rc == OK)
        {
           bytesWritten=0;
           char *buffer = orderIDs;

#ifdef SYBASE
           for(int districtIndex=1;districtIndex <=
DISTRICTS_PER_WAREHOUSE;districtIndex++)
           {
              if(dlvyTxn.out_dlvy.s_O_ID[districtIndex] == 0)
                 bytesWritten = sprintf(buffer,"\nD_ID %d had no
new orders",districtIndex);
              else
                 bytesWritten = sprintf(buffer,"%d
",dlvyTxn.out_dlvy.s_O_ID[districtIndex]);


              buffer+=bytesWritten;
           }
#else
           for(int districtIndex=0;districtIndex <
DISTRICTS_PER_WAREHOUSE;districtIndex++)
           {
              if(dlvyTxn.out_dlvy.s_O_ID[districtIndex] == 0)
                 bytesWritten = sprintf(buffer,"\nD_ID %d had no
new orders",districtIndex);
              else
                 bytesWritten = sprintf(buffer,"%d
",dlvyTxn.out_dlvy.s_O_ID[districtIndex]);


              buffer+=bytesWritten;
           }
#endif
        }
        else
           sprintf(orderIDs,"/nDelivery transaction failed");


        fprintf(dlvyLog,DELIVERY_LOG_SUCCESS_STR,
                dlvyCount,
                dlvyQueueData.enqueueTime.time,
                dlvyQueueData.enqueueTime.millitm,
                endQueueTime.time,
                endQueueTime.millitm,
```

```
                dlvyQueueData.warehouse,
                dlvyQueueData.in_s_0_CARRIER_ID,
                orderIDs,
                endProcessTime.time,
                endProcessTime.millitm);

        fflush(dlvyLog);
//    }
//    catch(...)
//    {
//       ERRORMSG("ERROR : Unhandled exeception in dlvy
thread. Thread exiting"<<endl);
//       fprintf(dlvyLog,"ERROR : Unhandled exeception in dlvy
thread %ld. Thread exiting.\n",GetCurrentThreadId());
//       fflush(dlvyLog);

//       LeaveCriticalSection(&dlvyQueueLock);
//       throw;
//    }
   }//end while true
}

/*
**********************************************************************
** Name         : queueDlvyTxn
** Description :
**           function queues dlvy txn in dlvy queue
** Parameters:
**           int    warehouse
**           short      carrier
** Returns      :
**           int    error code
** Comments     :
**            Function will queue dlvy txn if 2 points are true
**            1) We have room in our dlvy buffer
**            2) We writing over the end of the queue
**
**********************************************************************
*/

int queueDlvyTxn(int warehouse, short carrier_id)
{
   DEBUGMSG("Taking lock to queue dlvy txn.");

   EnterCriticalSection(&dlvyQueueLock);

   DEBUGMSG("Lock aquired to queue dlvy txn");

   if(dlvyBufferFreeSlots)
   {
      DEBUGMSG("Checking if we are inserting at tail of dlvy
queue."<<endl);
      if( dlvyBufferSlotIndex == (dlvyBufferThreadIndex-1))
      {
         ERRORMSG("Error dlvy queue inserting over
unserviced queued dlvy txn."<<endl);
         DEBUGMSG("Error dlvy queue inserting over
unserviced queued dlvy txn."<<endl);
         LeaveCriticalSection(&dlvyQueueLock);
         return ERR_DLVY_QUEUE_EATING_TAIL;
      }

      DEBUGMSG("free slots dlvy
queue:"<<dlvyBufferFreeSlots<<" inserting txn in slot: "
<<dlvyBufferSlotIndex<<
         "w_id: "<<warehouse<<" carrier: "<<carrier_id<<endl);

      dlvyQueue[dlvyBufferSlotIndex].warehouse = warehouse;
      dlvyQueue[dlvyBufferSlotIndex].in_s_0_CARRIER_ID =
carrier_id;

      _ftime(&dlvyQueue[dlvyBufferSlotIndex].enqueueTime);

//take lock here

      //decrement the number of free slots in the buffer
      dlvyBufferFreeSlots--;

      //increment the index to the next dlvy queue slot.
      dlvyBufferSlotIndex++;

      DEBUGMSG("dlvy txn queued, slots available in
queue:"<<dlvyBufferFreeSlots<<" queue slot
index:"<<dlvyBufferSlotIndex
            <<"w_id:"<<warehouse<<"
carrier:"<<carrier_id<<endl);

      DEBUGMSG("dlvy txn queued, slots available in queue:
"<<dlvyBufferFreeSlots<<" queue slot index:
"<<dlvyBufferSlotIndex
            <<" w_id: "<<warehouse<<" carrier:
"<<carrier_id<<endl);

      if(dlvyBufferSlotIndex == dlvyQueueLen)
      {
         DEBUGMSG("queue slot index hit end of queue, reset
to 0, current index:"<<dlvyBufferSlotIndex<<" free
slots:"<<dlvyBufferFreeSlots<<endl);
         ERRORMSG("queue slot index hit end of queue, reset
to 0, current index:"<<dlvyBufferSlotIndex<<" free
slots:"<<dlvyBufferFreeSlots<<
            "Thread Worker Queue
Index:"<<dlvyBufferThreadIndex<<endl);
         dlvyBufferSlotIndex=0;
      }
      //leave critical section
   }
   else
   {
      //no slots available in dlvy buffer, release critcal section
and return an nord->in_nord.in_item
      LeaveCriticalSection(&dlvyQueueLock);
      ERRORMSG("dlvy queue buffer full, increase the dlvy
queue length."<<endl);
      return ERR_DLVY_QUEUE_FULL;
   }

   LeaveCriticalSection(&dlvyQueueLock);

   //release semaphore to wake thread that there is work
   ReleaseSemaphore(dlvyThreadSemaphore,1,NULL);

   return OK;
}


/*
**********************************************************************
** Name         : doHtml
** Description :
**           HTML processing page entry point
** Parameters:
**           txn handle
** Returns      :
**           int - return code
** Comments     :
**
**********************************************************************
*/

void doHtml(TXN_HANDLE *txnHandle)
{
   DEBUGMSG("Entered doHtml(), parsing query string:"<<
txnHandle->urlString << " into command block"<< endl);
// ERRORMSG("Entered doHtml(), parsing query string:"<<
txnHandle->urlString << " into command block"<< endl);
   htmlPhrasercommandBlock(txnHandle->urlString);
   DEBUGMSG("Query string parsed.  command:"<<
commandBlock.getCommandId() << " user's terminal id:" <<
commandBlock.get_TERM_ID() << endl);

   int terminalID = atoi(commandBlock.get_TERM_ID());
   int commandID = commandBlock.getCommandId();

   DEBUGMSG("User sent in a terimal id:"<<terminalID<<",
checking to see if user has logged in before"<<endl);
```

```
   if(terminalID > 0)
   {
     DEBUGMSG("Terminal id > 0, user has logged in already,
terminalID:"<<terminalID<<" retrieving warehouse district
pair"<<endl);
      if(getTerminal(terminalID,txnHandle) != OK)
         return;
      DEBUGMSG("User had valid terminal id, user's login
warehouse:"<<txnHandle->w_id<<" district:"<<txnHandle-
>d_id<<endl);
   }
   else
   {
      DEBUGMSG("User did not submit a terminal id or valid
terminal id, ensure that the user is trying to log in."<<endl);
      if( (commandID != TXN_LOGIN) && (commandID !=
TXN_LOGIN_RESULTS) )
      {
         DEBUGMSG("ERROR : User has not logged
in."<<endl);
         ERRORMSG("ERROR : User has not logged
in."<<endl);
         sprintf(txnHandle->htmlPage,"ERROR : User has not
logged in or did not submit a valid terminal.");
         return;
      }
      DEBUGMSG("User is in process of logging in,
commandID:"<<commandID<<endl);
   }

   DEBUGMSG("Calling html page
function:"<<commandBlock.getCommandId()<<endl);
   int rc =
htmlPageFunctions[commandBlock.getCommandId()](&com
mandBlock,txnHandle);
   DEBUGMSG("Return from html page
function:"<<commandBlock.getCommandId()<<endl);

   return;
}

/*
**********************************************************************
** Name         : getTerminal
** Description :
**           retrieves terminal information based on terminal id
** Parameters:
**           int       terminal id
**           TERM_HANDLE*  txn handle
** Returns      :
**           int - return code
** Comments     :
**
**********************************************************************
*/
int getTerminal(int terminal,TXN_HANDLE *txnHandle)
{
// ERRORMSG(">>getTerminal"<<endl);
   //check to see if terminal id is out of range
   if(terminal >= numUsers)
   {
      //terminal id not valid.
      sprintf(txnHandle->htmlPage,"ERROR : Client does not
support more than %d users, terminal
id:%d",numUsers,terminal);
      ERRORMSG("ERROR : Client does not support more
than "<<numUsers<<" users, terminal id:"<<terminal<<endl);
      return ERR;
   }

   //check if terminal id is points to a not in use terminal
   if(!(termArray+terminal)->terminalInUse)
   {
      sprintf(txnHandle->htmlPage,"ERROR : Terminal id given
points to a not in use terminal.");
      ERRORMSG("ERROR :  Terminal id given points to a not
in use terminal."<<endl);
      return ERR;
   }

   DEBUGMSG("Storing terminal warehouse, district , and
initial term id for user:"<<terminal<<endl);

   //assign terminal values to txn_handle
   txnHandle->d_id = termArray[terminal].d_id;
   txnHandle->w_id = termArray[terminal].w_id;
   txnHandle->term_id = termArray[terminal].terminalID;

   DEBUGMSG("Users terminal:"<<terminal<<", stored
warehouse:"<<txnHandle->w_id<<
         " district:"<<txnHandle->d_id<<" termialID
stored:"<<txnHandle->term_id<<endl);

   return OK;
}

/*
**********************************************************************
** Name         : assignTerminal
** Description :
**           assigns terminal index to user
** Parameters:
**           TERM_HANDLE*  txn handle
** Returns      :
**           int - return code
** Comments     :
**
**********************************************************************
*/
int assignTerminal(TXN_HANDLE *txnHandle)
{
// ERRORMSG(">>assignTerminal"<<endl);
   EnterCriticalSection(&termLock);

   //check if terminal array is full.
   if(termNextFree == numUsers)
   {
      LeaveCriticalSection(&termLock);
      return ERR;
   }

   DEBUGMSG("Storing user warehouse:"<<txnHandle-
>w_id<<" district:"<< txnHandle->d_id<<
         " in terminal slot:"<<termNextFree<<endl);

   //store users w_id and d_id
   termArray[termNextFree].d_id = txnHandle->d_id;
   termArray[termNextFree].w_id = txnHandle->w_id;

   //set terminal slot to be in use
   termArray[termNextFree].terminalInUse = true;
   termArray[termNextFree].terminalID = termNextFree;
   //in txn handle, set the terminal id
   txnHandle->term_id = termNextFree;

   //increment to next free terminal.
   termNextFree++;

   DEBUGMSG("User warehouse:"<<txnHandle->w_id<<"
district:"<< txnHandle->d_id <<
      " stored in terminal slot:"<<txnHandle->term_id<<" next
terminal free:"<<termNextFree<<endl);

   LeaveCriticalSection(&termLock);

   return OK;
}
```

tpccIsapi.def

```
; tpccIsapi.def : declares the module parameters for the DLL.

LIBRARY      "tpccIsapi"

EXPORTS
   HttpExtensionProc
```

```
   GetExtensionVersion
   TerminateExtension
```

tpccIsapi.hpp

```
/*
********************************************************************
** Project    : AIX
** Component: Performance/TPC-W Benchmark
** Name         : tpccIsapi.hpp
** Title      : ISAPI interface for tpcc
********************************************************************
** Copyright (c) 2001,2002 IBM Corporation
** All rights reserved
********************************************************************
** History       :
**               Developed at IBM Austin by the AIX RS/6000
**               performance group.
**
** Comments      :
**
********************************************************************
*/

#ifndef __tpccISAPI_hpp__
#define __tpccISAPI_hpp__

#include <windows.h>
#include <httpext.h>

#include "tpcc.h"
#include "htmlPhraser.h"

#include <iomanip>

#ifdef DB2
   #include <db2tpcc.h>
#endif

#ifdef ORACLE
   #include <oratpcc.h>
#endif

#include <comsvcs.h>

//////////////////////////////////////////////////////////////////
// Terminal struct
//////////////////////////////////////////////////////////////////
struct TERM_ENTRY
{
   int     terminalID;
   bool   terminalInUse;
   int      w_id;
   short  d_id;
};

//////////////////////////////////////////////////////////////////
// COM interface
//////////////////////////////////////////////////////////////////
struct COM_HANDLE
{
   Itpcc_com   *comHandle;
   char      *txnBuffer;
   int         size;
};

//////////////////////////////////////////////////////////////////
// TXN handle
//////////////////////////////////////////////////////////////////
struct TXN_HANDLE
{
   char   htmlPage[MAX_HTML_PAGE_LEN];
   char   htmlHeader[MAX_HTML_HEADER_LEN];
   char   *urlString;

   //user data
   int      w_id;
   int      d_id;
   int      sync_id;
   int      term_id;
   int      conn_id;

   COM_HANDLE comInterface;
};

//////////////////////////////////////////////////////////////////
// Definitions
//////////////////////////////////////////////////////////////////
#define    INVALID_ITEM    100
#define    HEADER          "Content-
Type:text/html\r\nContent-Length: %d\r\nConnection: Keep-
Alive\r\n\r\n"
#define    TLS_NULL        0xFFFFFFFF
#define    ACCESS_TIMEOUT      3600000
   //One hour in milli seconds

#define    DELIVERY_LOG_SUCCESS_STR  "--Tran %d
Queue %d.%03d Start %d.%03d\nW_ID: %d CARRIER_ID:
%d %s\nend-time: %d.%03d\n"

//////////////////////////////////////////////////////////////////
// Function Prototypes
//////////////////////////////////////////////////////////////////
int initDlvy();
int initTxnHandle(TXN_HANDLE **txnHandle);
int closeTxnHandle(TXN_HANDLE *txnHandle);
int readRegisteryValues();
int getTerminal(int terminal,TXN_HANDLE *txnHandle);
int assignTerminal(TXN_HANDLE *txnHandle);
int getDBInstance();

void doHtml(TXN_HANDLE *txnHandle);
int doLoginForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doLoginResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doNewOrderForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doNewOrderResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doPaymentForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doPaymentResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doOrderStatusForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doOrderStatusResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doDeliveryForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doDeliveryResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doStockForm(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doStockResults(htmlPhraser
*commandBlock,TXN_HANDLE *txnHandle);
int doExit(htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle);

int doLoginErrorPage(char *htmlPage,char *message);
int doNewOrderErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle);
int doPaymentErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle);
int doOrderStatusErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle);
int doDeliveryErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle);
int doStockErrorPage(char *htmlPage,char
*message,htmlPhraser *commandBlock,TXN_HANDLE
*txnHandle);
```

```
void dlvyThreadEntry(void *);
int  queueDlvyTxn(int warehouse, short carrier_id);

int appendButtons(char *htmlPage);
int appendItems(char *htmlPage,short itemCount,short
cmdIDStart);
int appendHiddenFields(char *htmlPage,TXN_HANDLE
*txnHandle);

int displayStatus(char *htmlPage,int rc);

#endif
```

tpccIsapi.rc

```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "winres.h"
#include "atlsrvres.h"

//////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE
BEGIN
   "resource.h\0"
END

2 TEXTINCLUDE
BEGIN
   "#include ""winres.h""\r\n"
   "#include ""atlsrvres.h""\r\n"
   "\0"
END

3 TEXTINCLUDE
BEGIN
    "LANGUAGE LANG_ENGLISH,
SUBLANG_ENGLISH_US\r\n"
   "#pragma code_page(1252)\r\n"
   "#include ""atlsrv.rc""\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED



//////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x2L
 FILESUBTYPE 0x0L
BEGIN
   BLOCK "StringFileInfo"
   BEGIN
      BLOCK "040904e4"
      BEGIN
        VALUE "CompanyName", "TODO: <Company name>"
          VALUE "FileDescription", "TODO: <File description>"
          VALUE "FileVersion", "1.0.0.1"
          VALUE "InternalName", "isapi.dll"
          VALUE "LegalCopyright", "TODO: (c) <Company
name>.  All rights reserved."
          VALUE "OriginalFilename", "isapi.dll"
          VALUE "ProductName", "TODO: <Product name>"
          VALUE "ProductVersion", "1.0.0.1"
          VALUE "OLESelfRegister", ""
      END
   END
   BLOCK "VarFileInfo"
   BEGIN
     VALUE "Translation", 0x0409, 1252
   END
END



LANGUAGE 9, 1
#pragma code_page(1252)
//////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE
BEGIN
   IDS_PROJNAME           "tpccIsapi"
END

//////////////////////////////////////////////////////////////////////


#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#include "atlsrv.rc"

//////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

10.2. Client Transaction Code

dlvy.cpp

```
// tpccSybaseGlue.cpp : Defines the entry point for the dll
application
//
// This file implements code to translate and transport data to
and from
// the database layer, using Sybase ct-library calls.
//
//--------------------------------------------------------------------
// Klavs Pedersen     02-07-06 Separating delivery transaction
out due to
//                        optimization issues.
//

#include "stdafx.h"
```

```
#include "tpccSybaseGlue.h"
#include "SybUtils.h"

extern "C" int do_rtn(void *ctx,CS_COMMAND *cmd,CS_INT
*val);


void init_dlvy(void *ctx)
{
   syb_con_data_t    *stx = (syb_con_data_t *)ctx;
   CS_INT          rc;

   //allocate data fields
   ERRORMSG(">>init_dlvy"<<endl);
   DTAFMT(stx-
>p_delP.in.datafmt_w_id,CS_SMALLINT_TYPE,"@w_id");
   DTAFMT(stx-
>p_delP.in.datafmt_d_id,CS_TINYINT_TYPE,"@d_id");
   DTAFMT(stx-
>p_delP.in.datafmt_carrier_id,CS_SMALLINT_TYPE,"@o_ca
rrier_id");

   DTAFMT_BND(stx-
>p_delP.out.datafmt_o_id,CS_INT_TYPE,CS_INT,1);

   ERRORMSG("Allocating cmd"<<endl);
   RTN_ON_FAIL(stx->context,
      ct_cmd_alloc(stx->conn, &stx->del_cmd),
   "Could not allocate command structure.");

   RTN_ON_FAIL(stx->context,
      ct_cmd_props(stx->del_cmd, CS_SET,
CS_STICKY_BINDS, &stx->sticky_binds, CS_UNUSED,
NULL),
    "init_dlvy: ct_cmd_props() failed.");

   CT_CMD(stx->del_cmd,CS_RPC_CMD, "delivery");
   ERRORMSG("initialized delivery stored procedure"<<endl);

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->del_cmd, &stx-
>p_delP.in.datafmt_w_id,&stx->delP.in_dlvy.s_W_ID, &stx-
>delP.in_dlvy.w_id_len, &stx->nonullind),
   "ct_setparam() for @w_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->del_cmd, &stx-
>p_delP.in.datafmt_d_id,&stx->delP.in_dlvy.s_D_ID, &stx-
>delP.in_dlvy.d_id_len, &stx->nonullind),
   "ct_setparam() for @d_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->del_cmd, &stx-
>p_delP.in.datafmt_carrier_id, &(stx-
>delP.in_dlvy.s_O_CARRIER_ID), &stx-
>delP.in_dlvy.carrier_id_len, &stx->nonullind),
   "ct_setparam() for @carrier_id failed.");

   /*
   CS_INT buf = CS_TRUE;
   if (ct_cmd_props(stx-
>del_cmd,CS_SET,CS_STICKY_BINDS,(CS_VOID
*)&buf,CS_UNUSED,NULL) != CS_SUCCEED)
   {
      ERRORMSG("[init_dlvy]error from ct_cmd_props()"<<
endl);
   }
   */
   ERRORMSG("<<init_dlvy"<<endl);
}

/*
*********************************************************************
** Name         :  do_dlvy
** Description :
**            Function calls db2 api to execute ords txn
** Parameters:
**            dlvy_wrapper*   dlvy txn structs wrapper
**            void*        stored context
** Returns      :
**            int - return code
** Comments     :
**            Attach to thread's context, call nord sql function
**            then detach from context.
*********************************************************************
*/

extern "C" TPCCSYBASEGLUE_API int
do_dlvy(dlvy_wrapper *dlvy,void *ctx)
{

   syb_con_data_t    *stx = (syb_con_data_t *)ctx;
   CS_INT          rc,dist=1;
    int deadlock = 0;
   int count = 0;
   long result_type = 0;
   int terror = 0,num_cols = 0;
   CS_RETCODE     ret=CS_ROW_FAIL;

   DEBUGMSG2(">>do_dlvy"<<endl);
   memcpy(&stx->delP,dlvy,sizeof(dlvy_wrapper));
   stx->delP.in_dlvy.w_id_len = CS_SIZEOF(stx-
>delP.in_dlvy.s_W_ID);
   stx->delP.in_dlvy.d_id_len = CS_SIZEOF(stx-
>delP.in_dlvy.s_D_ID);
   stx->delP.in_dlvy.carrier_id_len = CS_SIZEOF(stx-
>delP.in_dlvy.s_O_CARRIER_ID);
// memset(&stx->delP.out_dlvy.s_O_ID,0,sizeof(stx-
>delP.out_dlvy.s_O_ID))


   DEBUGMSG2(stx->delP.in_dlvy.s_W_ID<<" "<<
      stx->delP.in_dlvy.s_D_ID<<" "<<
      stx->delP.in_dlvy.s_O_CARRIER_ID<<"
dist="<<dist<<endl);

   JUST_RETURN( ct_send(stx->del_cmd), "ct_send()
failed.");

   DEBUGMSG2("back from send"<<endl);


   /*
   ** Process the results.  Loop while ct_results() returns
CS_SUCCEED.
   */
   while ((rc = ct_results(stx->del_cmd, &result_type)) ==
CS_SUCCEED)
   {
      switch ((int)result_type)
      {
      CS_INT v;
      case CS_END_RESULTS:
         ERRORMSG("CS_END_RESULTS???"<<endl);
         ERRORMSG(stx->delP.in_dlvy.s_W_ID<<" "<<
            stx->delP.in_dlvy.s_D_ID<<" "<<
            stx->delP.in_dlvy.s_O_CARRIER_ID<<endl);
         if (do_rtn(ctx,stx->del_cmd,&v) != CS_SUCCEED)
         {
            ERRORMSG("[Delivery]: do_rtn() failed"<<endl);
            return ret;
         }
//       (CS_VOID)ct_cancel((CS_CONNECTION *)NULL, stx-
>del_cmd, CS_CANCEL_ALL);
         ret = CS_END_DATA;
         break;
      case CS_CMD_SUCCEED:
         DEBUGMSG2("CS_CMD_SUCCEED"<<endl);
         /*
         ** This means no rows were returned.
         */
         break;

      case CS_CMD_DONE:
         DEBUGMSG2("CS_CMD_DONE"<<endl);
```

```
        /*
        ** This means we're done with one result set.
        */
        break;

      case CS_PARAM_RESULT:
        ERRORMSG("CS_PARAM_RESULT"<<endl);
        if (do_rtn(ctx,stx->del_cmd,&v) != CS_SUCCEED)
        {
          ERRORMSG("[Delivery]: do_rtn() failed"<<endl);
          return ret;
        }
        if (v != 0)
        {
          terror = 1;
          ERRORMSG("[DY]Transaction error, return
code="<<v);

        }
        break;
      case CS_STATUS_RESULT:
        DEBUGMSG2("CS_STATUS_RESULT"<<endl);
        if (do_rtn(ctx,stx->del_cmd,&v) != CS_SUCCEED)
        {
          ERRORMSG("[Delivery]: do_rtn() failed"<<endl);
          return ret;
        }
        break;
      case CS_CMD_FAIL:
        DEBUGMSG2("CS_CMD_FAIL"<<endl);
        /*
        ** This means that the server encountered an error
while
        ** processing our command.
        */
        ERRORMSG("[Delivery] ct_results() returned
CMD_FAIL"<<endl);
        terror = 1;
        break;

      case CS_ROW_RESULT:
        DEBUGMSG2("CS_ROW_RESULT"<<endl);
        //bind to the low count db var

        DEBUGMSG2("district set to:"<<dist<<endl);

        CT_BND(stx->del_cmd,1,stx->p_delP.out.datafmt_o_id,
          stx->delP.out_dlvy.s_O_ID[dist]);
        DEBUGMSG2("Bound to stx->p_delP.out.datafmt_o_id
"<<"value is: "<<stx->delP.out_dlvy.s_O_ID[dist]<<endl);

        while ((ret = ct_fetch(stx-
>del_cmd,CS_UNUSED,CS_UNUSED,CS_UNUSED,NULL))
== CS_SUCCEED)
        {
          if (dist > 10)
          {
            ERRORMSG("[DL] Too many districts."<<endl);
            return -1;
          }
          DEBUGMSG2("s_O_ID: "<<stx-
>delP.out_dlvy.s_O_ID[dist]<<endl);
          dist++;
          DEBUGMSG2("Incrementing district
to:"<<dist<<endl);

//          CT_BND(stx->del_cmd,1,stx-
>p_delP.out.datafmt_o_id,
//                stx->delP.out_dlvy.s_O_ID[dist]);

        }
        DEBUGMSG2("ct_fetch, ret="<<ret<<endl);
      }
    }

    if (ret != CS_END_DATA)
    {
//    (CS_VOID)ct_cmd_drop(stx->cmd);
      (CS_VOID)ct_cancel((CS_CONNECTION *)NULL, stx-
>del_cmd, CS_CANCEL_ALL);
      ERRORMSG("[Delivery] ret != CS_END_RESULTS,
ret="<<ret<<", rc="<<rc<<endl);
      terror = 1;
    }

    stx->delP.out_dlvy.s_transtatus = terror == 0 ? 0 : -1;

  memcpy(dlvy,&stx->delP,sizeof(dlvy_wrapper));
  DEBUGMSG2("<<do_dlvy"<<endl);
  return OK;
}
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard
includes
// tpccSybaseGlue.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

// TODO: reference any additional headers you need in
STDAFX.H
// and not in this file
```

stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently, but
// are changed infrequently
//

#pragma once

#define WIN32_LEAN_AND_MEAN     // Exclude rarely-used
stuff from Windows headers
// Windows Header Files:
#include <windows.h>

// TODO: reference additional headers your program requires
here
```

sybtpcc.h

```
// Common defines and structures use internally by client
code
// Not to be confused with structures actually passed in
transaxtions
//
#ifndef _SYBTPCC_H_
#define _SYBTPCC_H_
#define MAX_DISTINCTS     500

#include <cspublic.h>
#include <ctpublic.h>
#include <ctpublic.h>

struct in_neword_type {

  CS_INT i_id_len;
  CS_INT qty_len;
  CS_INT s_w_id_len;
  CS_INT w_id_len;
  CS_INT d_id_len;
  CS_INT c_id_len;
  CS_INT ol_cnt_len;
  CS_INT  s_OL_I_ID[15];
  CS_INT  s_OL_SUPPLY_W_ID[15];
  CS_INT  s_OL_QUANTITY[15];
  CS_INT  s_C_ID;
  CS_INT  s_W_ID;
  CS_INT  s_D_ID;
  CS_INT  s_O_OL_CNT;
  CS_INT  s_all_local;
  CS_INT  duplicate_items;
```

```
  CS_DATETIME cr_date;
};

struct out_neword_type {
  double  s_I_PRICE[16];
  double  s_OL_AMOUNT[16];
  CS_INT  s_S_QUANTITY[16];
  CS_CHAR     s_I_NAME[16][25];
  CS_CHAR     s_brand_generic[16][2];
  CS_CHAR  s_O_ENTRY_D_time[22];
  CS_DATETIME  s_O_ENTRY_D_base;
  CS_CHAR  b_g[16][22];
  float  s_W_TAX;
  float  s_D_TAX;
  float  s_C_DISCOUNT;
  double  s_total_amount;
  CS_INT  s_O_ID;
  CS_INT  s_O_OL_CNT;
  CS_SMALLINT  s_transtatus;
  CS_SMALLINT  deadlocks;
  CS_SMALLINT  datelen;
  CS_CHAR     s_C_LAST[17];
  CS_CHAR     s_C_CREDIT[3];
  CS_INT  terror;
  CS_INT  status;
  CS_INT  retry;
  CS_SMALLINT  items_valid;
};

struct in_payment_struct {
  CS_INT w_id_len;
  CS_INT c_w_id_len;
  CS_INT h_amount_cents_len;
  CS_INT d_id_len;
  CS_INT c_d_id_len;
  CS_INT c_id_len;
  CS_INT c_last_len;
  double  s_H_AMOUNT;
  CS_SMALLINT  s_W_ID;
  CS_SMALLINT  s_C_W_ID;
  CS_INT  s_C_ID;
  CS_TINYINT  s_C_D_ID;
  CS_TINYINT  s_D_ID;
  CS_CHAR s_C_LAST[17];
  CS_INT  bylastname;
  CS_DATETIME cr_date;
};

struct out_payment_struct {
  CS_CHAR  s_H_DATE_time[22];
  CS_CHAR  c_since[22];
  double  s_C_CREDIT_LIM;
  double  s_C_BALANCE;
  float  s_C_DISCOUNT;
  CS_INT  s_C_ID;
  CS_SMALLINT  s_transtatus;
  CS_SMALLINT  since_len;
  CS_INT  terror;
  CS_SMALLINT  hlen;
  CS_INT  retry;
  CS_CHAR     s_W_STREET_1[21];
  CS_CHAR     s_W_STREET_2[21];
  CS_CHAR     s_W_CITY[21];
  CS_CHAR     s_W_STATE[3];
  CS_CHAR     s_W_ZIP[10];
  CS_CHAR     s_D_STREET_1[21];
  CS_CHAR     s_D_STREET_2[21];
  CS_CHAR     s_D_CITY[21];
  CS_CHAR     s_D_STATE[3];
  CS_CHAR     s_D_ZIP[10];
  CS_CHAR     s_C_FIRST[17];
  CS_CHAR     s_C_MIDDLE[3];
  CS_CHAR     s_C_LAST[17];
  CS_CHAR     s_C_STREET_1[21];
  CS_CHAR     s_C_STREET_2[21];
  CS_CHAR     s_C_CITY[21];
  CS_CHAR     s_C_STATE[3];
  CS_CHAR     s_C_ZIP[10];
  CS_CHAR     s_C_PHONE[17];
  CS_CHAR     s_C_CREDIT[3];
  CS_CHAR     s_C_DATA[201];
};

struct in_ordstat_struct {
  CS_INT  s_C_ID;
  CS_INT  s_W_ID;
  CS_INT  s_D_ID;
  CS_INT  bylastname;
  CS_CHAR     s_C_LAST[17];
  CS_INT w_id_len;
  CS_INT d_id_len;
  CS_INT c_id_len;
  CS_INT c_last_len;
};

struct out_ordstat_struct {
  double   s_C_BALANCE;
  char     s_O_ENTRY_D_time[22];
  CS_INT  s_C_ID;
  CS_INT  s_O_ID;
  CS_INT  s_O_CARRIER_ID;
  CS_INT  s_ol_cnt;
  CS_INT  terror;
  char     s_OL_DELIVERY_D_time[16][26];
  CS_DATETIME  s_OL_DELIVERY_D_base[16];

  double  s_OL_AMOUNT[16];
  CS_INT  s_OL_I_ID[16];
  CS_INT  s_OL_SUPPLY_W_ID[16];
  CS_INT  s_OL_QUANTITY[16];
  CS_INT  ol_del_len[16];
  CS_SMALLINT  s_transtatus;
  CS_INT  retry;
  CS_CHAR     s_C_FIRST[17];
  CS_CHAR     s_C_MIDDLE[3];
  CS_CHAR     s_C_LAST[17];
};

struct in_delivery_struct {
   CS_INT w_id_len;
   CS_INT d_id_len;
   CS_INT carrier_id_len;
  double  s_O_DELIVERY_D_time;
  CS_INT  s_W_ID;
  CS_INT  s_D_ID;
  CS_INT  s_O_CARRIER_ID;
  CS_SMALLINT  plsqlflag;
  CS_DATETIME cr_date;
};

struct out_delivery_struct {
  CS_INT  s_O_ID[12];
  CS_INT  terror;
  CS_SMALLINT s_transtatus;
  CS_INT  retry;
};

struct in_stocklev_struct{
   CS_INT   s_threshold;
   CS_INT   s_D_ID;
   CS_INT   s_W_ID;
   CS_INT   w_id_len;
   CS_INT   d_id_len;
   CS_INT   threshold_len;

};

struct out_stocklev_struct{
   CS_INT   s_low_stock;
   CS_INT   s_transtatus;
   CS_INT   deadlocks;
};
#endif //_SYBTPCC_H_
```

sybUtils.cpp

```
/*
**
** Description
** -----------
**    This example file contains utility routines which are used
by all other
**    sample programs. It demonstrates how an application
can hide some of the
**    implementation details of CT-Lib from higher level
programs.
**
** Routines Used
** -------------
**    cs_ctx_alloc
**
*/

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <cspublic.h>
#include <ctpublic.h>
#include "SybUtils.h"

#if !lint
static char Sccsid[] = {"%Z% %M% %I% %G%"};
#endif /* !lint */

/*********************************************************************
********
**
** display functions
**
*********************************************************************
*******/

/*
** ex_display_dlen()
**
** Type of function:
** example program api
**
** Purpose:
**
** Returns:
** Nothing.
**
** Side Effects:
** None
**
** History:
** 11/12/92: Otto Lind : Created
*/

CS_INT CS_PUBLIC
ex_display_dlen(CS_DATAFMT       *column)
{
  CS_INT   len;

  switch ((int) column->datatype)
  {
    case CS_CHAR_TYPE:
    case CS_LONGCHAR_TYPE:
    case CS_VARCHAR_TYPE:
    case CS_TEXT_TYPE:
    case CS_IMAGE_TYPE:
      len = MIN(column->maxlength, MAX_CHAR_BUF);
      break;

    case CS_UNICHAR_TYPE:
     len = MIN((column->maxlength / 2), MAX_CHAR_BUF);
      break;

    case CS_BINARY_TYPE:
    case CS_VARBINARY_TYPE:
      len = MIN((2 * column->maxlength) + 2,
MAX_CHAR_BUF);
      break;

    case CS_BIT_TYPE:
    case CS_TINYINT_TYPE:
      len = 3;
      break;

    case CS_SMALLINT_TYPE:
      len = 6;
      break;

    case CS_INT_TYPE:
      len = 11;
      break;

    case CS_REAL_TYPE:
    case CS_FLOAT_TYPE:
      len = 20;
      break;

    case CS_MONEY_TYPE:
    case CS_MONEY4_TYPE:
      len = 24;
      break;

    case CS_DATETIME_TYPE:
    case CS_DATETIME4_TYPE:
      len = 30;
      break;

    case CS_NUMERIC_TYPE:
    case CS_DECIMAL_TYPE:
      len = (CS_MAX_PREC + 2);
      break;

    default:
      len = 12;
      break;
  }

  return MAX((CS_INT)(strlen(column->name) + 1), len);
}

/*
** ex_display_header()
**
** Type of function:
** example program api
**
** Purpose:
**
** Returns:
** Nothing.
**
** Side Effects:
** None
**
** History:
** 11/12/92: Otto Lind : Created
*/

CS_RETCODE CS_PUBLIC
ex_display_header(CS_INT    numcols, CS_DATAFMT
columns[])
{
  CS_INT   i;
  CS_INT   l;
  CS_INT   j;
  CS_INT   disp_len;

  fputc('\n', stdout);
  for (i = 0; i < numcols; i++)
  {
    disp_len = ex_display_dlen(&columns[i]);
    fprintf(stdout, "%s", columns[i].name);
    fflush(stdout);
    l = disp_len - strlen(columns[i].name);
```

```
    for (j = 0; j < l; j++)
    {
      fputc(' ', stdout);
      fflush(stdout);
    }
  }
  fputc('\n', stdout);
  fflush(stdout);
  for (i = 0; i < numcols; i++)
  {
    disp_len = ex_display_dlen(&columns[i]);
    l = disp_len - 1;
    for (j = 0; j < l; j++)
    {
      fputc('-', stdout);
    }
    fputc(' ', stdout);
  }
  fputc('\n', stdout);

  return CS_SUCCEED;
}

/*
** ex_display_column()
**
** Type of function:
** example program api
**
** Purpose:
**
** Returns:
** Nothing.
**
** Side Effects:
** None
**
** History:
** 11/12/92: Otto Lind : Created
*/

CS_RETCODE CS_PUBLIC
ex_display_column(CS_CONTEXT  *context, CS_DATAFMT
  *colfmt, CS_VOID   *data, CS_INT     datalength, CS_INT
  indicator)
{
  char     *null = "NULL";
  char     *nc   = "NO CONVERT";
  char     *cf   = "CONVERT FAILED";
  CS_DATAFMT  srcfmt;
  CS_DATAFMT  destfmt;
  CS_INT   olen;
  CS_CHAR     wbuf[MAX_CHAR_BUF];
  CS_BOOL     res;
  CS_INT   i;
  CS_INT   disp_len;
  CS_SMALLINT indi;

  indi = (CS_SMALLINT)indicator;

  if (indi == CS_NULLDATA)
  {
    olen = strlen(null);
    strcpy(wbuf, null);
  }
  else
  {
    cs_will_convert(context, colfmt->datatype,
CS_CHAR_TYPE, &res);
    if (res != CS_TRUE)
    {
      olen = strlen(nc);
      strcpy(wbuf, nc);
    }
    else
    {
      srcfmt.datatype  = colfmt->datatype;
      srcfmt.format    = colfmt->format;
      srcfmt.locale    = colfmt->locale;
      srcfmt.maxlength = datalength;

      memset(&destfmt, 0, sizeof(destfmt));
      memset(wbuf, 0, MAX_CHAR_BUF);

      destfmt.maxlength = MAX_CHAR_BUF;
      destfmt.datatype  = CS_CHAR_TYPE;
      destfmt.format    = CS_FMT_NULLTERM;
      destfmt.locale    = NULL;

      if (cs_convert(context, &srcfmt, data, &destfmt,
          wbuf, &olen) != CS_SUCCEED)
      {
        olen = strlen(cf);
        strcpy(wbuf, cf);
      }
      else
      {
        /*
        ** output length include null
        ** termination
        */
        olen -= 1;
      }
    }
  }

  for (i = 0; i < olen; i++)
  {
    fprintf(stdout, "%c", wbuf[i]);
  }

  fflush(stdout);

  disp_len = ex_display_dlen(colfmt);
  for (i = 0; i < (disp_len - olen); i++)
  {
    fputc(' ', stdout);
  }
  fflush(stdout);

  return CS_SUCCEED;
}

/*****************************************************************
********
**
** error functions
**
*****************************************************************
*******/

/*
** ex_panic()
**
** Type of function:
**    example program utility api
**
** Purpose:
**    Reports a string message to EX_ERROR_OUT, and
exits program.
**
** Returns:
**    nothing
**
** Side Effects:
**    Terminates program
*/

CS_VOID CS_PUBLIC
ex_panic(char*msg)
{
  fprintf(EX_ERROR_OUT, "ex_panic: FATAL ERROR:
%s\n", msg);
  fflush(EX_ERROR_OUT);
  exit(EX_EXIT_FAIL);
}
```

```
/****************************************************************************
********
**
** callback functions
**
****************************************************************************
*******/

/*
** ex_clientmsg_cb()
**
** Type of function:
**    example program client message handler
**
** Purpose:
**    Installed as a callback into Open Client.
**
** Returns:
**    CS_SUCCEED
**
** Side Effects:
**    None
*/

CS_RETCODE CS_PUBLIC
ex_clientmsg_cb(CS_CONTEXT   *context,
CS_CONNECTION *connection, CS_CLIENTMSG   *errmsg)
{
   fprintf(EX_ERROR_OUT, "\nOpen Client Message:\n");
  fprintf(EX_ERROR_OUT, "Message number: LAYER = (%d)
ORIGIN = (%d) ",
      CS_LAYER(errmsg->msgnumber), CS_ORIGIN(errmsg-
>msgnumber));
   fprintf(EX_ERROR_OUT, "SEVERITY = (%d) NUMBER =
(%d)\n",
      CS_SEVERITY(errmsg->msgnumber),
CS_NUMBER(errmsg->msgnumber));
   fprintf(EX_ERROR_OUT, "Message String: %s\n", errmsg-
>msgstring);
   if (errmsg->osstringlen > 0)
   {
      fprintf(EX_ERROR_OUT, "Operating System Error:
%s\n",
         errmsg->osstring);
   }
   fflush(EX_ERROR_OUT);

   return CS_SUCCEED;
}

/*
** ex_servermsg_cb()
**
** Type of function:
**    example program server message handler
**
** Purpose:
**    Installed as a callback into Open Client.
**
** Returns:
**    CS_SUCCEED
**
** Side Effects:
**    None
*/
CS_RETCODE CS_PUBLIC
ex_servermsg_cb(CS_CONTEXT *context,
CS_CONNECTION *connection, CS_SERVERMSG *srvmsg)
{
   fprintf(EX_ERROR_OUT, "\nServer message:\n");
   fprintf(EX_ERROR_OUT, "Message number: %d, Severity
%d, ",
      srvmsg->msgnumber, srvmsg->severity);
   fprintf(EX_ERROR_OUT, "State %d, Line %d\n",
      srvmsg->state, srvmsg->line);

   if (srvmsg->svrnlen > 0)
   {
      fprintf(EX_ERROR_OUT, "Server '%s'\n", srvmsg-
>svrname);
   }

   if (srvmsg->proclen > 0)
   {
      fprintf(EX_ERROR_OUT, " Procedure '%s'\n", srvmsg-
>proc);
   }

   fprintf(EX_ERROR_OUT, "Message String: %s\n", srvmsg-
>text);
   fflush(EX_ERROR_OUT);

   return CS_SUCCEED;
}

/****************************************************************************
********
**
** utility functions
**
****************************************************************************
*******/

/*
** ex_init()
**
** Type of function:
**    example program utility api
**
** Purpose:
**    This function allocates the context structure, initializes
**Client-Library, and installs the default callbacks. If this
function
**should fail, it will deallocated all memory allocations it has
done.
**
**The callbacks are installed at the context level. Other
applications
**may need to install callbacks at the connection level.
**
** Parameters:
**    context     - Pointer to A Pointer to CS_CONTEXT
structure.
**
** Returns:
**    Result of initialization functions from CT-Lib.
**
*/

CS_RETCODE CS_PUBLIC
ex_init(CS_CONTEXT   **context)
{
   CS_RETCODE retcode;
   CS_INT   netio_type = CS_SYNC_IO;

   /*
   ** Get a context handle to use.
   */
   retcode = cs_ctx_alloc(EX_CTLIB_VERSION, context);
   if (retcode != CS_SUCCEED)
   {
      ex_error("ex_init: cs_ctx_alloc() failed");
      return retcode;
   }

   /*
   ** Initialize Open Client.
   */
   retcode = ct_init(*context, EX_CTLIB_VERSION);
   if (retcode != CS_SUCCEED)
   {
      ex_error("ex_init: ct_init() failed");
```

```
    cs_ctx_drop(*context);
    *context = NULL;
    return retcode;
  }

#ifdef EX_API_DEBUG
  /*
  ** ct_debug stuff. Enable this function right before any call
to
  ** OC-Lib that is returning failure.
  */
  retcode = ct_debug(*context, NULL, CS_SET_FLAG,
CS_DBG_API_STATES,
          NULL, CS_UNUSED);
  if (retcode != CS_SUCCEED)
  {
    ex_error("ex_init: ct_debug() failed");
  }
#endif

  /*
  ** Install client and server message handlers.
  */
  if (retcode == CS_SUCCEED)
  {
    retcode = ct_callback(*context, NULL, CS_SET,
CS_CLIENTMSG_CB,
            (CS_VOID *)ex_clientmsg_cb);
    if (retcode != CS_SUCCEED)
    {
      ex_error("ex_init: ct_callback(clientmsg) failed");
    }
  }

  if (retcode == CS_SUCCEED)
  {
    retcode = ct_callback(*context, NULL, CS_SET,
CS_SERVERMSG_CB,
          (CS_VOID *)ex_servermsg_cb);
    if (retcode != CS_SUCCEED)
    {
      ex_error("ex_init: ct_callback(servermsg) failed");
    }
  }

  /*
  ** This is an synchronous example so set the input/output
type
  ** to synchronous (This is the default setting, but show an
  ** example anyway).
  */
  if (retcode == CS_SUCCEED)
  {
    retcode = ct_config(*context, CS_SET, CS_NETIO,
&netio_type,
                CS_UNUSED, NULL);
    if (retcode != CS_SUCCEED)
    {
      ex_error("ex_init: ct_config(netio) failed");
    }
  }

  if (retcode != CS_SUCCEED)
  {
    ct_exit(*context, CS_FORCE_EXIT);
    cs_ctx_drop(*context);
    *context = NULL;
  }

  return retcode;
}

/*
** ex_connect()
**
** Type of function:
**    example program utility api
**
** Purpose:
**    This routine establishes a connection to the server
identified
**    in example.h and sets the CS_USER, CS_PASSWORD,
and
**    CS_APPNAME properties for the connection.
**
**    If a connection property is NULL, it is not set.
**
**    If this function should fail, it will deallocated all memory
**    allocations it has done.
**
** Parameters:
**    context     - Pointer to CS_CONTEXT structure.
**    connection  - Pointer to CS_CONNECTION pointer.
**appname     - Name of application calling this routine.
**username- user name to use when connecting.
**password - password to use when connecting.
**server      - server to connect to.
**
** Return:
**    Result of function calls from CT-Lib.
*/

CS_RETCODE CS_PUBLIC
ex_connect(CS_CONTEXT  *context, CS_CONNECTION
  **connection, CS_CHAR      *appname, CS_CHAR
  *username, CS_CHAR     *password, CS_CHAR
  *server)
{
      CS_INT     len;
  CS_RETCODE retcode;
  CS_BOOL     hafailover = CS_TRUE;

  /*
  ** Allocate a connection structure.
  */
  retcode = ct_con_alloc(context, connection);
  if (retcode != CS_SUCCEED)
  {
    ex_error("ct_con_alloc failed");
    return retcode;
  }

  /*
  ** If a username is defined, set the CS_USERNAME
property.
  */
  if (retcode == CS_SUCCEED && username != NULL)
  {
    if ((retcode = ct_con_props(*connection, CS_SET,
CS_USERNAME,
          username, CS_NULLTERM, NULL)) !=
CS_SUCCEED)
    {
      ex_error("ct_con_props(username) failed");
    }
  }

  /*
  ** If a password is defined, set the CS_PASSWORD
property.
  */
  if (retcode == CS_SUCCEED && password != NULL)
  {
    if ((retcode = ct_con_props(*connection, CS_SET,
CS_PASSWORD,
          password, CS_NULLTERM, NULL)) !=
CS_SUCCEED)
    {
      ex_error("ct_con_props(password) failed");
    }
  }

  /*
  ** Set the CS_APPNAME property.
  */
```

```
   if (retcode == CS_SUCCEED && appname != NULL)
   {
      if ((retcode = ct_con_props(*connection, CS_SET,
CS_APPNAME,
            appname, CS_NULLTERM, NULL)) !=
CS_SUCCEED)
      {
         ex_error("ct_con_props(appname) failed");
      }
   }

#ifdef HAFAILOVER

   /*
   ** Set the CS_HAFAILOVER property.
   */
   if (retcode == CS_SUCCEED )
   {
      if ((retcode = ct_con_props(*connection, CS_SET,
CS_HAFAILOVER,
            &hafailover, CS_UNUSED, NULL)) !=
CS_SUCCEED)
      {
         ex_error("ct_con_props(CS_HAFAILOVER) failed");
      }
   }
#endif  /* HAFAILOVER */


   /*
   ** Open a Server connection.
   */
   if (retcode == CS_SUCCEED)
   {
      len = (server == NULL) ? 0 : CS_NULLTERM;
      retcode = ct_connect(*connection, server, len);
      if (retcode != CS_SUCCEED)
      {
         ex_error("ct_connect failed");
      }
   }

   if (retcode != CS_SUCCEED)
   {
      ct_con_drop(*connection);
      *connection = NULL;
   }
   return retcode;
}

/*
** ex_con_cleanup()
**
** Type of function:
**    example program utility api
**
** Purpose:
**    The routine closes a connection and deallocates the
**CS_CONNECTION structure.
**
** Parameters:
**    connection  - Pointer to connection structure.
**    status    - status of last interaction with this Client-Library
**          If not ok, this routine will perform a force close.
**
** Returns:
**    Result of function calls from CT-Lib.
*/

CS_RETCODE CS_PUBLIC
ex_con_cleanup(CS_CONNECTION   *connection,
CS_RETCODE status)
{
   CS_RETCODE retcode;
   CS_INT    close_option;

   close_option = (status != CS_SUCCEED) ?
CS_FORCE_CLOSE : CS_UNUSED;
   retcode = ct_close(connection, close_option);
   if (retcode != CS_SUCCEED)
   {
      ex_error("ex_con_cleanup: ct_close() failed");
      return retcode;
   }
   retcode = ct_con_drop(connection);
   if (retcode != CS_SUCCEED)
   {
      ex_error("ex_con_cleanup: ct_con_drop() failed");
      return retcode;
   }

   return retcode;
}

/*
** ex_ctx_cleanup()
**
** Type of function:
**    example program utility api
**
** Purpose:
**    The routine exits Client-Library deallocates the
**CS_CONTEXT structure.
**
** Parameters:
**    context     - Pointer to context structure.
**    status    - status of last interaction with Client-Library.
**          If not ok, this routine will perform a force exit.
**
** Returns:
**    Result of function calls from CT-Lib.
*/

CS_RETCODE CS_PUBLIC
ex_ctx_cleanup(CS_CONTEXT *context, CS_RETCODE
   status)
{
   CS_RETCODE retcode;
   CS_INT    exit_option;

   exit_option = (status != CS_SUCCEED) ?
CS_FORCE_EXIT : CS_UNUSED;
   retcode = ct_exit(context, exit_option);
   if (retcode != CS_SUCCEED)
   {
      ex_error("ex_ctx_cleanup: ct_exit() failed");
      return retcode;
   }
   retcode = cs_ctx_drop(context);
   if (retcode != CS_SUCCEED)
   {
      ex_error("ex_ctx_cleanup: cs_ctx_drop() failed");
      return retcode;
   }
   return retcode;
}

/*
** ex_execute_cmd()
**
** Type of function:
**      example program utility api
**
** Purpose:
**      This routine sends a language command to the server.
It expects no
**rows or parameters to be returned from the server.
**
** Parameters:
**      connection       - Pointer to CS_COMMAND structure.
**      cmdbuf          - The buffer containing the command.
**
** Return:
**      Result of functions called in CT-Lib
*/
```

```
CS_RETCODE CS_PUBLIC
ex_execute_cmd(CS_CONNECTION   *connection,
CS_CHAR        *cmdbuf)
{
      CS_RETCODE      retcode;
      CS_INT          restype;
      CS_COMMAND      *cmd;
      CS_RETCODE      query_code;

      /*
      ** Get a command handle, store the command string in
it, and
      ** send it to the server.
      */
      if ((retcode = ct_cmd_alloc(connection, &cmd)) !=
CS_SUCCEED)
      {
            ex_error("ex_execute_cmd: ct_cmd_alloc() failed");
            return retcode;
      }

      if ((retcode = ct_command(cmd, CS_LANG_CMD,
cmdbuf, CS_NULLTERM,
                  CS_UNUSED)) != CS_SUCCEED)
      {
            ex_error("ex_execute_cmd: ct_command() failed");
            (void)ct_cmd_drop(cmd);
            return retcode;
      }

      if ((retcode = ct_send(cmd)) != CS_SUCCEED)
      {
            ex_error("ex_execute_cmd: ct_send() failed");
            (void)ct_cmd_drop(cmd);
            return retcode;
  }

      /*
      ** Examine the results coming back. If any errors are
seen, the query
      ** result code (which we will return from this function)
will be
      ** set to FAIL.
      */
      query_code = CS_SUCCEED;
      while ((retcode = ct_results(cmd, &restype)) ==
CS_SUCCEED)
      {
            switch((int)restype)
            {
               case CS_CMD_SUCCEED:
               case CS_CMD_DONE:
                  break;

               case CS_CMD_FAIL:
                  query_code = CS_FAIL;
                  break;

               case CS_STATUS_RESULT:
                  retcode = ct_cancel(NULL, cmd,
CS_CANCEL_CURRENT);
                  if (retcode != CS_SUCCEED)
                  {
                        ex_error("ex_execute_cmd: ct_cancel()
failed");
                        query_code = CS_FAIL;
                  }
                  break;

               default:
                  /*
                  ** Unexpected result type.
                  */
                  query_code = CS_FAIL;
                  break;
            }
            if (query_code == CS_FAIL)
            {
                  /*
                  ** Terminate results processing and break out
of
                  ** the results loop
                  */
                  retcode = ct_cancel(NULL, cmd,
CS_CANCEL_ALL);
                  if (retcode != CS_SUCCEED)
                  {
                        ex_error("ex_execute_cmd: ct_cancel()
failed");
                  }
                  break;
            }
      }

      /*
      ** Clean up the command handle used
      */
      if (retcode == CS_END_RESULTS)
      {
            retcode = ct_cmd_drop(cmd);
            if (retcode != CS_SUCCEED)
            {
                  query_code = CS_FAIL;
            }
      }
      else
      {
            (void)ct_cmd_drop(cmd);
            query_code = CS_FAIL;
      }

      return query_code;
}

/*
** ex_fetch_data()
**
** Type of function:
**    example program utility api
**
** Purpose:
**    This function processes fetchable results sets. The
results include:
**
**    CS_ROW_RESULT
**    CS_CURSOR_RESULT
**    CS_PARAM_RESULT
**    CS_STATUS_RESULT
**    CS_COMPUTE_RESULT
**
** Since the Client-Library result model has been unified, the
same
** apis are used for each of the above result types.
**
**One caveat is the processing of CS_COMPUTE_RESULTs.
The name field
** sent from the server is typically zero length. To display a
meaningful
** header, the aggregate compute operator name should be
found for the
** column, and that name used instead. The compute
example program has
** code which demonstrates this.
**
** Parameters:
** cmd - Pointer to command structure
**
** Return:
** CS_MEM_ERROR  If an memory allocation failed.
** CS_SUCCEED If the data was displayed.
** CS_FAIL     If no columns were present.
**    <retcode>   Result of the Client-Library function if a
failure was
**       returned.
**
*/
```

```
CS_RETCODE CS_PUBLIC
ex_fetch_data(CS_COMMAND  *cmd,CS_INT *val)
{
   CS_RETCODE    retcode;
   CS_INT      num_cols;
   CS_INT      i;
   CS_INT      j;
   CS_INT      row_count = 0;
   CS_INT      rows_read;
   CS_INT      disp_len;
   CS_DATAFMT    *datafmt;
   EX_COLUMN_DATA    *coldata;

   /*
   ** Find out how many columns there are in this result set.
   */
   retcode = ct_res_info(cmd, CS_NUMDATA, &num_cols,
CS_UNUSED, NULL);
   if (retcode != CS_SUCCEED)
   {
      ex_error("ex_fetch_data: ct_res_info() failed");
      return retcode;
   }

   /*
   ** Make sure we have at least one column
   */
   if (num_cols <= 0)
   {
      ex_error("ex_fetch_data: ct_res_info() returned zero
columns");
      return CS_FAIL;
   }

   /*
   ** Our program variable, called 'coldata', is an array of
   ** EX_COLUMN_DATA structures. Each array element
represents
   ** one column.  Each array element will re-used for each
row.
   **
   ** First, allocate memory for the data element to process.
   */
   coldata = (EX_COLUMN_DATA *)malloc(num_cols * sizeof
(EX_COLUMN_DATA));
   if (coldata == NULL)
   {
      ex_error("ex_fetch_data: malloc() failed");
      return CS_MEM_ERROR;
   }

   datafmt = (CS_DATAFMT *)malloc(num_cols * sizeof
(CS_DATAFMT));
   if (datafmt == NULL)
   {
      ex_error("ex_fetch_data: malloc() failed");
      free(coldata);
      return CS_MEM_ERROR;
   }

   /*
   ** Loop through the columns getting a description of each
one
   ** and binding each one to a program variable.
   **
   ** We're going to bind each column to a character string;
   ** this will show how conversions from server native
datatypes
   ** to strings can occur via bind.
   **
   ** We're going to use the same datafmt structure for both
the describe
   ** and the subsequent bind.
   **
   ** If an error occurs within the for loop, a break is used to
get out
   ** of the loop and the data that was allocated is free'd
before
   ** returning.
   */
   for (i = 0; i < num_cols; i++)
   {
      /*
      ** Get the column description.  ct_describe() fills the
      ** datafmt parameter with a description of the column.
      */
      retcode = ct_describe(cmd, (i + 1), &datafmt[i]);
      if (retcode != CS_SUCCEED)
      {
         ex_error("ex_fetch_data: ct_describe() failed");
         break;
      }

      /*
      ** update the datafmt structure to indicate that we want
the
      ** results in a null terminated character string.
      **
      ** First, update datafmt.maxlength to contain the
maximum
      ** possible length of the column. To do this, call
      ** ex_display_len() to determine the number of bytes
needed
      ** for the character string representation, given the
      ** datatype described above.  Add one for the null
      ** termination character.
      */
      datafmt[i].maxlength = ex_display_dlen(&datafmt[i]) + 1;

      /*
      ** Set datatype and format to tell bind we want things
      ** converted to null terminated strings
      */
      datafmt[i].datatype = CS_CHAR_TYPE;
      datafmt[i].format   = CS_FMT_NULLTERM;

      /*
      ** Allocate memory for the column string
      */
      coldata[i].value = (CS_CHAR
*)malloc(datafmt[i].maxlength);
      if (coldata[i].value == NULL)
      {
         ex_error("ex_fetch_data: malloc() failed");
         retcode = CS_MEM_ERROR;
         break;
      }

      /*
      ** Now bind.
      */
      retcode = ct_bind(cmd, (i + 1), &datafmt[i],
            coldata[i].value, &coldata[i].valuelen,
            (CS_SMALLINT *)&coldata[i].indicator);
      if (retcode != CS_SUCCEED)
      {
         ex_error("ex_fetch_data: ct_bind() failed");
         break;
      }
   }
   if (retcode != CS_SUCCEED)
   {
      for (j = 0; j < i; j++)
      {
         free(coldata[j].value);
      }
      free(coldata);
      free(datafmt);
      return retcode;
   }

   /*
   ** Display column header
   */
// ex_display_header(num_cols, datafmt);
```

```
   /*
   ** Fetch the rows.  Loop while ct_fetch() returns
CS_SUCCEED or
   ** CS_ROW_FAIL
   */
   while (((retcode = ct_fetch(cmd, CS_UNUSED,
CS_UNUSED, CS_UNUSED,
         &rows_read)) == CS_SUCCEED) || (retcode ==
CS_ROW_FAIL))
   {
      /*
      ** Increment our row count by the number of rows just
fetched.
      */
      row_count = row_count + rows_read;

      /*
      ** Check if we hit a recoverable error.
      */
      if (retcode == CS_ROW_FAIL)
      {
         ex_error("Error on row");
      }

      /*
      ** We have a row.  Loop through the columns displaying
the
      ** column values.
      */
      *val = atoi(coldata[0].value);
/*
      for (i = 0; i < num_cols; i++)
      {
         // Display the column value
         fprintf(stdout, "%s", coldata[i].value);
         fflush(stdout);


         // If not last column, Print out spaces between this
         // column and next one.

         if (i != num_cols - 1)
         {
            disp_len = ex_display_dlen(&datafmt[i]);
            disp_len -= coldata[i].valuelen - 1;
            for (j = 0; j < disp_len; j++)
            {
               fputc(' ', stdout);
            }
         }
      }
      fprintf(stdout, "\n");
      fflush(stdout);
*/
   }
   /*
   ** Free allocated space.
   */
   for (i = 0; i < num_cols; i++)
   {
      free(coldata[i].value);
   }
   free(coldata);
   free(datafmt);

   /*
   ** We're done processing rows.  Let's check the final return
   ** value of ct_fetch().
   */
   switch ((int)retcode)
   {
      case CS_END_DATA:
         /*
         ** Everything went fine.
         */
//       fprintf(stdout, "All done processing rows.\n");
//       fflush(stdout);
         retcode = CS_SUCCEED;
         break;

      case CS_FAIL:
         /*
         ** Something terrible happened.
         */
         ex_error("ex_fetch_data: ct_fetch() failed");
         return retcode;
         /*NOTREACHED*/
         break;

      default:
         /*
         ** We got an unexpected return value.
         */
         ex_error("ex_fetch_data: ct_fetch() returned an
expected retcode");
         return retcode;
         /*NOTREACHED*/
         break;

   }

   return retcode;
}


/*****************************************************************
********
**
** sql based functions
**
*****************************************************************
*******/

/*
** ex_create_db()
**
** Type of function:
**      example program utility api
**
** Purpose:
**      This routine creates a database and opens it. It first
checks
**      that the database does not already exists. If it does
exist
**      the database is dropped before creating a new one.
**
** Parameters:
**      connection      - Pointer to CS_CONNECTION structure.
**      dbname          - The name to be used for the created
database.
**
** Return:
**      Result of functions called in CT-Lib.
**
*/

CS_RETCODE CS_PUBLIC
ex_create_db(CS_CONNECTION   *connection, char
*dbname)
{
      CS_RETCODE      retcode;
      CS_CHAR         *cmdbuf;

      /*
      ** If the database already exists, drop it.
      */
      if ((retcode = ex_remove_db(connection, dbname)) !=
CS_SUCCEED)
      {
            ex_error("ex_create_db: ex_remove_db() failed");
      }

      /*
      ** Allocate the buffer for the command string.
      */
      cmdbuf = (CS_CHAR *) malloc(EX_BUFSIZE);
```

```
    if (cmdbuf == (CS_CHAR *)NULL)
    {
        ex_error("ex_create_db: malloc() failed");
        return CS_FAIL;
    }

    /*
    ** Set up and send the command to create the database.
    */
    sprintf(cmdbuf, "create database %s", dbname);
    if ((retcode = ex_execute_cmd(connection, cmdbuf)) !=
CS_SUCCEED)
    {
        ex_error("ex_create_db: ex_execute_cmd(create
db) failed");
    }
  free(cmdbuf);
  return retcode;
}

/*
** ex_remove_db()
**
** Type of function:
**     example program utility api
**
** Purpose:
**     This routine removes a database. It first checks that
**     the database exists, and if so, removes it.
**
** Parameters:
**     connection     - Pointer to CS_CONNECTION structure.
**     dbname         - The name of the database to remove.
**
** Return:
**     Result of functions called in CT-Lib or CS_FAIL if a
malloc failure
**     occurred.
*/

CS_RETCODE
ex_remove_db(CS_CONNECTION   *connection, char
*dbname)
{
    CS_RETCODE     retcode;
    CS_CHAR        *cmdbuf;

    /*
    ** Connect to the master database in order to
    ** remove the specified database.
    */
    if ((retcode = ex_use_db(connection, "master")) !=
CS_SUCCEED)
    {
        ex_error("ex_remove_db: ex_use_db(master)
failed");
        return retcode;
    }

    /*
    ** Allocate the buffer for the command string.
    */
    cmdbuf = (CS_CHAR *) malloc(EX_BUFSIZE);
    if (cmdbuf == (CS_CHAR *)NULL)
    {
        ex_error("ex_remove_db: malloc() failed");
        return CS_FAIL;
    }

    /*
    ** Set up and send the command to check for and drop
the
    ** database if it exists.
    */
    sprintf(cmdbuf,
        "if exists (select name from sysdatabases where
name = \"%s\") \
        drop database %s", dbname, dbname);
    if ((retcode = ex_execute_cmd(connection, cmdbuf)) !=
CS_SUCCEED)
    {
        ex_error("ex_remove_db: ex_execute_cmd(drop
db) failed");
    }
    free(cmdbuf);
    return retcode;
}

/*
** ex_use_db()
**
** Type of function:
**     example program utility api
**
** Purpose:
**     This routine changes the current database to the named
db passed in.
**
** Parameters:
**     connection     - Pointer to CS_CONNECTION structure.
**     dbname         - The name of the database to use.
**
** Return:
**     Result of functions called in CT-Lib or CS_FAIL if a
malloc failure
**     occured.
*/

CS_RETCODE
ex_use_db(CS_CONNECTION   *connection, char
*dbname)
{
    CS_RETCODE     retcode;
    CS_CHAR        *cmdbuf;

    /*
    ** Allocate the buffer for the command string.
    */
    cmdbuf = (CS_CHAR *) malloc(EX_BUFSIZE);
    if (cmdbuf == (CS_CHAR *)NULL)
    {
        ex_error("ex_use_db: malloc() failed");
        return CS_FAIL;
    }

    /*
    ** Set up and send the command to use the database
    */
  sprintf(cmdbuf, "use %s\n", dbname);
    if ((retcode = ex_execute_cmd(connection, cmdbuf)) !=
CS_SUCCEED)
    {
        ex_error("ex_use_db: ex_execute_cmd(use db)
failed");
    }
    free(cmdbuf);
    return retcode;
}
```

SybUtils.h

```
/*
** exutils.h
**
** Header file which contains the defines and prototypes for
the utility
** functions in exutils.c
**
*/

/* Sccsid %Z% %M% %I% %G% */

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

```
** defines and typedefs used
**
*****************************************************************************
*******/
#ifndef _SYBUTILS_H_
#define _SYBUTILS_H_
#ifndef MAX
#define MAX(X,Y) (((X) > (Y)) ? (X) : (Y))
#endif

#ifndef MIN
#define MIN(X,Y)  (((X) < (Y)) ? (X) : (Y))
#endif

/*
** Maximum character buffer for displaying a column
*/
#define MAX_CHAR_BUF1024

/*
** Define structure where row data is bound.
*/
typedef struct _ex_column_data
{
   CS_INT    indicator;
   CS_CHAR     *value;
   CS_INT    valuelen;
} EX_COLUMN_DATA;

/*****************************************************************************
********
**
** protoypes for all public functions
**
*****************************************************************************
*******/
/* exutils.c */
extern CS_INT CS_PUBLIC ex_display_dlen PROTOTYPE((
   CS_DATAFMT *column
   ));
extern CS_RETCODE CS_PUBLIC ex_display_header
PROTOTYPE((
   CS_INT numcols,
   CS_DATAFMT columns[]
   ));
extern CS_RETCODE CS_PUBLIC ex_display_column
PROTOTYPE((
   CS_CONTEXT *context,
   CS_DATAFMT *colfmt,
   CS_VOID *data,
   CS_INT datalength,
   CS_INT indicator
   ));
extern CS_VOID CS_PUBLIC ex_panic PROTOTYPE((
   char *msg
   ));
extern CS_VOID CS_PUBLIC ex_error PROTOTYPE((
   char *msg
   ));
extern CS_RETCODE CS_PUBLIC ex_clientmsg_cb
PROTOTYPE((
   CS_CONTEXT *context,
   CS_CONNECTION *connection,
   CS_CLIENTMSG *errmsg
   ));
extern CS_RETCODE CS_PUBLIC ex_servermsg_cb
PROTOTYPE((
   CS_CONTEXT *context,
   CS_CONNECTION *connection,
   CS_SERVERMSG *srvmsg
   ));
extern CS_RETCODE CS_PUBLIC ex_init PROTOTYPE((
   CS_CONTEXT **context
   ));
extern CS_RETCODE CS_PUBLIC ex_connect
PROTOTYPE((
   CS_CONTEXT *context,
   CS_CONNECTION **connection,
   CS_CHAR *appname,
   CS_CHAR *username,
   CS_CHAR *password,
   CS_CHAR *server
   ));
extern CS_RETCODE CS_PUBLIC ex_con_cleanup
PROTOTYPE((
   CS_CONNECTION *connection,
   CS_RETCODE status
   ));
extern CS_RETCODE CS_PUBLIC ex_ctx_cleanup
PROTOTYPE((
   CS_CONTEXT *context,
   CS_RETCODE status
   ));
extern CS_RETCODE CS_PUBLIC ex_execute_cmd
PROTOTYPE((
   CS_CONNECTION *connection,
   CS_CHAR *cmdbuf
   ));
extern CS_RETCODE CS_PUBLIC ex_fetch_data
PROTOTYPE((
   CS_COMMAND *cmd,
   CS_INT *val
   ));
extern CS_RETCODE CS_PUBLIC ex_create_db
PROTOTYPE((
   CS_CONNECTION *connection,
   char *dbname
   ));
extern CS_RETCODE ex_remove_db PROTOTYPE((
   CS_CONNECTION *connection,
   char *dbname
   ));
extern CS_RETCODE ex_use_db PROTOTYPE((
   CS_CONNECTION *connection,
   char *dbname
   ));
/* Sccsid %Z% %M% %I% %G% */

/*
** Define symbolic names, constants, and macros
*/
#define EX_MAXSTRINGLEN    255
#define EX_BUFSIZE    1024
#define EX_CTLIB_VERSION CS_VERSION_125
#define EX_BLK_VERSION    BLK_VERSION_150
#define EX_ERROR_OUT      stderr

/*
** exit status values
*/
#ifdef vms
#include <stsdef.h>
#define EX_EXIT_SUCCEED    (STS$M_INHIB_MSG |
STS$K_SUCCESS)
#define EX_EXIT_FAIL     (STS$M_INHIB_MSG |
STS$K_ERROR)
#else
#define EX_EXIT_SUCCEED    0
#define EX_EXIT_FAIL     1
#endif /* vms */

/*
** Define global variables used in all sample programs
*/
#define EX_SERVER NULL    /* use DSQUERY env var */
#define EX_USERNAME  "sa"
#define EX_PASSWORD  ""

/*
** Uncomment the following line to test the HA Failover
feature.
*/
/* #define HAFAILOVER 1 */

/*
```

```
** For some platforms (e.g. windows 3.1), additional work
needs to be done
** to insure that the output of some of the example programs
can be displayed.
** This macro will insure that any setup is done for the
platform.
**
** For windows, _wsetscreenbuf(_fileno(stdout),
_WINBUFINT) will set
** QuickWin's standard output screen buffer to unlimited size.
*/

#if QWIN

#define EX_SCREEN_INIT() _wsetscreenbuf(_fileno(stdout),
_WINBUFINF)

#else /* QWIN */
#endif
#define EX_SCREEN_INIT()
#endif // _SYBUTILS_H_
```

tpccSybaseGlue.cpp

```
// tpccSybaseGlue.cpp : Defines the entry point for the dll
application
//
// This file implements code to translate and transport data to
and from
// the database layer, using Sybase ct-library calls.
//
//-------------------------------------------------------------------------
// Joe Noyola     09-09-05 Create File
// Klavs Pedersen 10-25-05 Translate transaction code from
db-lib to ct-lib
// Klavs Pedersen 10-29-05 Optimizating, only call command
allocation and
//                         initialization once at startup, use
CS_STICKY_BINDS.
// Klavs Pedersen 04-06-06 Audited
#include "stdafx.h"
#include "tpccSybaseGlue.h"
#include "SybUtils.h"

void *conHandle;

static char *days[] = {"Mon ", "Tue ", "Wed ", "Thu ", "Fri ",
"Sat ", "Sun "};
static char *mths[] = {"Jan ", "Feb ", "Mar ", "Apr ", "May ",
"Jun ", "Jul ",
                "Aug ", "Sep ", "Oct ", "Nov ", "Dec "};
static char *tofd[] = {"AM ","PM "};


extern void init_dlvy(void *ctx);

ofstream debugStream;
ofstream errorStream;

CRITICAL_SECTION debugMutex;
CRITICAL_SECTION errorMutex;
FILE *respTimes;
HANDLE                  ASemaphore;

char userName[16] = {NULL};
char userPassword[16] = {NULL};
char dbName[32] = {"tpcc"};
#ifdef DEBUG
int debugFlag = 1;
#else
int debugFlag = 0;
#endif

int count = 0;
CS_CONTEXT     *global_context;


#define INVALID_ITEM 100


void convert_dt(CS_CONTEXT *ctx,CS_DATETIME *Date,
char *str)
{
   CS_DATEREC date_rec;

   DEBUGMSG("Before cs_dt_crack"<<endl);
   cs_dt_crack(ctx,CS_DATETIME_TYPE,(CS_VOID
*)Date,&date_rec);
   DEBUGMSG("After cs_dt_crack"<<endl);
   sprintf(str,"%s%02d %02d:%02d:%02d %4d",

   mths[date_rec.datemonth],date_rec.datedmonth,date_rec.d
atehour,

   date_rec.dateminute,date_rec.datesecond,date_rec.dateye
ar);
}

void sort_order_lines (struct in_neword_type *nord)
{
   /*
   ** Sort order_lines in a new_order by i_id. Reduces
possibility of deadlock.
   ** Brute force insertion sort -- works OK for <= 15 rows.
   */
   int            i, j;
   CS_INT      i_temp,w_temp,q_temp;

   for (j=1; j<nord->s_O_OL_CNT; j++) {
      if (nord->s_OL_I_ID[j-1] > nord->s_OL_I_ID[j]) {
         i_temp = nord->s_OL_I_ID[j];
         w_temp = nord->s_OL_SUPPLY_W_ID[j];
         q_temp = nord->s_OL_QUANTITY[j];
         nord->s_OL_I_ID[j] = nord->s_OL_I_ID[j-1];
         nord->s_OL_SUPPLY_W_ID[j] = nord-
>s_OL_SUPPLY_W_ID[j-1];
         nord->s_OL_QUANTITY[j] = nord->s_OL_QUANTITY[j-
1];
         for (i=j-2; i>=0 && i_temp < nord->s_OL_I_ID[i]; i--) {
            nord->s_OL_I_ID[i+1] = nord->s_OL_I_ID[i];
            nord->s_OL_SUPPLY_W_ID[i+1] = nord-
>s_OL_SUPPLY_W_ID[i];
            nord->s_OL_QUANTITY[i+1] = nord-
>s_OL_QUANTITY[i];
         }
         nord->s_OL_I_ID[i+1] = i_temp;
         nord->s_OL_SUPPLY_W_ID[i+1] = w_temp;
         nord->s_OL_QUANTITY[i+1] = q_temp;

      }
   }
}

CS_RETCODE CS_PUBLIC
cserror_cb(CS_CONTEXT     *context, CS_CLIENTMSG
*errmsg)
{
   ERRORMSG("Message number: LAYER =
("<<CS_LAYER(errmsg->msgnumber)<<
      ") ORIGIN = ("<<CS_ORIGIN(errmsg->msgnumber)<<
      ") SEVERITY = ("<<CS_SEVERITY(errmsg-
>msgnumber)<<
      ") NUMBER = ("<<CS_NUMBER(errmsg-
>msgnumber)<<")"<<endl);
   ERRORMSG("Message: "<<errmsg->msgstring<<endl);
   if (errmsg->osstringlen > 0)
   {
      ERRORMSG("Operating System Error: "<<
         errmsg->osstring<<endl);
   }
   return CS_SUCCEED;
}

CS_RETCODE CS_PUBLIC
servermsg_cb(CS_CONTEXT     *context,
CS_CONNECTION  *connection, CS_SERVERMSG
*srvmsg)
```

```
{
   ERRORMSG("Message number: "<<srvmsg-
>msgnumber<<
      ", Severity "<<srvmsg->severity<<
      ", State "<<srvmsg->state<<", Line "<<srvmsg-
>line<<endl);

   if (srvmsg->svrnlen > 0)
   {
      ERRORMSG("Server '"<<srvmsg->svrname<<"'"<<endl);
   }

   if (srvmsg->proclen > 0)
   {
      ERRORMSG(" Procedure '"<<srvmsg->proc<<"'"<<endl);
   }

   ERRORMSG("Message: "<<srvmsg->text<<endl);

   return CS_SUCCEED;
}

CS_RETCODE CS_PUBLIC
clientmsg_cb(CS_CONTEXT     *context, CS_CONNECTION
*connection, CS_CLIENTMSG   *errmsg)
{
   /*
   ** Error number:
   ** Print the error's severity, number, origin, and layer.
   ** These four numbers uniquely identify the error.
   */
   ERRORMSG("Client Library error: severity: "<<(long)
CS_SEVERITY(errmsg->severity)<<
      " number: "<<(long) CS_NUMBER(errmsg-
>msgnumber)<<
      " origin: "<<(long) CS_ORIGIN(errmsg->msgnumber)<<
      " layer: "<<(long) CS_LAYER(errmsg-
>msgnumber)<<endl);
   /*
   ** Error text:
   ** Print the error text.
   */
   ERRORMSG( errmsg->msgstring<<endl);
   /*
   ** Operating system error information:
   ** Some errors, such as network errors, may have
   ** an operating system error associated with them.
   ** If there was an operating system error,
   ** this code prints the error message text.
   */
   if (errmsg->osstringlen > 0)
   {
      ERRORMSG("Operating system error number("<<(long)
errmsg->osnumber<<"): "<<
         errmsg->osstring<<endl);
   }
   /*
   ** If we return CS_FAIL, Client-Library marks the
connection
   ** as dead. This means that it cannot be used anymore.
   ** If we return CS_SUCCEED, the connection remains
alive
   ** if it was not already dead.
   */
   return (CS_SUCCEED);
}

BOOL APIENTRY DllMain( HANDLE hModule,
                       DWORD  ul_reason_for_call,
                       LPVOID lpReserved
                     )
{
   HKEY   registryKey;
DWORD  regType;
char   value[MAX_STRING_LEN];
char   msgstr[4096] = {NULL};
DWORD  regValueSize = MAX_STRING_LEN;

   switch (ul_reason_for_call)
   {
   case DLL_PROCESS_ATTACH:

      ASemaphore = CreateSemaphore(NULL, 0, 100, NULL);

      if(debugFlag)
      {
         InitializeCriticalSection(&debugMutex);

         debugStream.rdbuf( )-
>open("C:\\inetpub\\wwwroot\\tpcc\\debug_gluecode.txt",ios_
base::out | ios_base::app );
         if(!debugStream.rdbuf( ) ->is_open())
            return FALSE;
      }

      DEBUGMSG("Entered dllMain of tpccSybaseGlue.dll" <<
endl);
      InitializeCriticalSection(&errorMutex);
      errorStream.rdbuf( )-
>open("C:\\inetpub\\wwwroot\\tpcc\\error_glueScode.txt",
ios_base::out | ios_base::trunc);

      if(!errorStream.rdbuf( )->is_open())
         return FALSE;

      ERRORMSG("Error log opened."<<endl);
#ifdef TIMING

   respTimes=fopen("c:\\inetpub\\wwwroot\\tpcc\\respTimes","
wb");
      if(!respTimes)
      {
         ERRORMSG("Unable to open response time file
c:\\inetpub\\wwwroot\\tpcc\\respTimes"<<endl);
         return FALSE;
      }
      ERRORMSG("Response time file created:"<<endl);
#endif


      DEBUGMSG("Opening registery sub key "<<
REGISTERY_SUB_KEY << endl);
      //open up registery key

   if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTERY
_SUB_KEY,0,KEY_READ,&registryKey) ==
ERROR_SUCCESS)
      {
         DEBUGMSG("Registery key open"<<endl);
         //get the null db user name
         regValueSize = sizeof(value);
         if
(RegQueryValueEx(registryKey,DB_USER_NAME,0,&regTyp
e,(BYTE *) &value,&regValueSize)== ERROR_SUCCESS )
            strcpy(userName,value);
         else
            return ERR_INVALID_USERNAME;
         DEBUGMSG("DB user name:"<< userName << endl);

         regValueSize = sizeof(value);
         if
(RegQueryValueEx(registryKey,DB_USER_PASSWORD,0,&
regType,(BYTE *) &value,&regValueSize)==
ERROR_SUCCESS )
            strcpy(userPassword,value);
         else
            return ERR_INVALID_PASSWORD;
         DEBUGMSG("DB user password:"<<userPassword <<
endl);

         regValueSize = sizeof(value);
         if
(RegQueryValueEx(registryKey,DB_NAME,0,&regType,(BYT
E *) &value,&regValueSize)== ERROR_SUCCESS )
            strcpy(dbName,value);
         else
```

```
            return ERR_INVALID_DB_NAME;
         RegCloseKey(registryKey);
         DEBUGMSG("DB name:"<<dbName << endl);

      }
      else
      {
         return ERR_INVALID_REGISTERY_KEY;
         DEBUGMSG("Unable to open registery key"<<
REGISTERY_SUB_KEY << endl);
      }

      break;
   case DLL_THREAD_ATTACH:
      break;
   case DLL_THREAD_DETACH:
      break;
   case DLL_PROCESS_DETACH:
      #ifdef TIMING
         ERRORMSG("dll_process_detach called, closing
timing file"<<endl);
         fclose(respTimes);
      #endif
      ERRORMSG("dll_process_detach called"<<endl);
//    disconnect_db(conHandle);
      break;
   }


    return TRUE;
}

/*
****************************************************************
** Name         :  attachContext
** Description :
**             Function calls db2 api to attach thread to
**             a specific context per thread basis.
** Parameters:
**             void*     stored context
** Returns      :
**             int - return code
** Comments     :
**
****************************************************************
*/
extern "C" int attachContext(void *ctx)
{
   return OK;
}

/*
****************************************************************
** Name         :  detachContext
** Description :
**             Function calls db2 api to detach thread from
context
** Parameters:
**             void*     stored context
** Returns      :
**             int - return code
** Comments     :
**
****************************************************************
*/
extern "C" int detachContext(void *ctx)
{
   return OK;
}

/*
** ex_error()
**
** Type of function:
**    example program utility api
**
** Purpose:
**    Reports a string message to EX_ERROR_OUT.
**
** Returns:
**    nothing
**
** Side Effects:
**    none.
*/

CS_VOID CS_PUBLIC
ex_error(char *msg)
{
   ERRORMSG( "ERROR: "<<msg<<endl);
}

extern "C" void init_rtn(void *ctx)
{
   syb_con_data_t    *stx = (syb_con_data_t *)ctx;

   memset(&stx->datafmt_rtn,0,sizeof(stx->datafmt_rtn));

   stx->datafmt_rtn.datatype = 0;
   stx->datafmt_rtn.format = CS_FMT_NULLTERM;
   stx->datafmt_rtn.maxlength = 12;
}

extern "C" int do_rtn(void *ctx,CS_COMMAND *cmd,CS_INT
*val)
{
   syb_con_data_t    *stx = (syb_con_data_t *)ctx;
   CS_RETCODE     retcode;
   CS_INT       num_cols;
   CS_INT       row_count = 0;
   CS_INT       rows_read;
   CS_INT       v_len=12;
   CS_CHAR         v[12];

   if ((retcode = ct_res_info(cmd, CS_NUMDATA, &num_cols,
CS_UNUSED, NULL)) != CS_SUCCEED)
   {
      ex_error("ex_fetch_data: ct_res_info() failed");
      return retcode;
   }

   if (num_cols != 1)
   {
      ex_error("do_rtn: ct_res_info() Did not return 1 column");
      return CS_FAIL;
   }
   if ((retcode = ct_bind(cmd, 1, &stx->datafmt_rtn,v,
&v_len,(CS_SMALLINT *)NULL)) != CS_SUCCEED)
   {
      ex_error("do_rtn: ct_bind() failed");
      return CS_FAIL;
   }
   while (((retcode = ct_fetch(cmd, CS_UNUSED,
CS_UNUSED, CS_UNUSED,
         &rows_read)) == CS_SUCCEED) || (retcode ==
CS_ROW_FAIL))
   {
      /*
      ** Increment our row count by the number of rows just
fetched.
      */
      row_count = row_count + rows_read;

      /*
      ** Check if we hit a recoverable error.
      */
      if (retcode == CS_ROW_FAIL)
      {
         ex_error("Error on row");
      }

      /*
      ** We have a row.  Loop through the columns displaying
the
      ** column values.
```

```
      */
      *val = atoi(v);
   }
   return CS_SUCCEED;
}

extern "C" void init_nord(void *ctx)
{
   int i;
   char p_i_id[10], p_qty[10],buf[10],p_s_w_id[10];
   syb_con_data_t    *stx = (syb_con_data_t *)ctx;
   CS_INT         rc;

   ERRORMSG(">>init_nord"<<endl);
   //allocate a new command structure
   RTN_ON_FAIL(stx->context,
   ct_cmd_alloc(stx->conn, &stx->nor_local_cmd),
   "Could not allocate command structure.");

   RTN_ON_FAIL(stx->context,
   ct_cmd_alloc(stx->conn, &stx->nor_remote_cmd),
   "Could not allocate command structure.");

   ERRORMSG("allocated cmds"<<endl);
   //Init rpc
// CT_CMD(stx->nor_local_cmd,CS_RPC_CMD,
"neworder_local_debug1");
// CT_CMD(stx->nor_remote_cmd,CS_RPC_CMD,
"neworder_remote_debug1");

   //prepare in data fields
   DTAFMT(stx-
>p_norP.in.datafmt_w_id,CS_SMALLINT_TYPE,"@w_id");
   DTAFMT(stx-
>p_norP.in.datafmt_d_id,CS_TINYINT_TYPE,"@d_id");
   DTAFMT(stx-
>p_norP.in.datafmt_c_id,CS_INT_TYPE,"@c_id");
   DTAFMT(stx-
>p_norP.in.datafmt_ol_cnt,CS_INT_TYPE,"@o_ol_cnt");

   strcpy(p_i_id,"@i_id");strcpy(p_qty,"@ol_qty");strcpy(p_s_
w_id,"@s_w_id");
   DTAFMT(stx-
>p_norP.in.ol[0].datafmt_i_id,CS_INT_TYPE,p_i_id);
   DTAFMT(stx-
>p_norP.in.ol[0].datafmt_qty,CS_INT_TYPE,p_qty);
   DTAFMT(stx-
>p_norP.in.ol[0].datafmt_supply_w_id,CS_SMALLINT_TYPE,
p_s_w_id);
   for (i=1;i<15;i++)
   {
      strcpy(&p_i_id[5],itoa(i+1,buf,10));
      strcpy(&p_qty[7],itoa(i+1,buf,10));
      strcpy(&p_s_w_id[7],itoa(i+1,buf,10));
      DTAFMT(stx-
>p_norP.in.ol[i].datafmt_supply_w_id,CS_SMALLINT_TYPE,
p_s_w_id);
      DTAFMT(stx-
>p_norP.in.ol[i].datafmt_i_id,CS_INT_TYPE,p_i_id);
      DTAFMT(stx-
>p_norP.in.ol[i].datafmt_qty,CS_SMALLINT_TYPE,p_qty);
   }

   //prepare out data fields
   DTAFMT_CHAR_BND(stx-
>p_norP.out.ol.datafmt_i_name,CS_CHAR_TYPE,1,25);
   DTAFMT_BND(stx-
>p_norP.out.ol.datafmt_i_price,CS_FLOAT_TYPE,CS_FLOA
T,2);
   DTAFMT_BND(stx-
>p_norP.out.ol.datafmt_s_quantity,CS_SMALLINT_TYPE,CS
_SMALLINT,3);
   DTAFMT_CHAR_BND(stx-
>p_norP.out.ol.datafmt_b_g,CS_CHAR_TYPE,4,2);
   DTAFMT_BND(stx-
>p_norP.out.datafmt_w_tax,CS_REAL_TYPE,CS_REAL,1);
   DTAFMT_BND(stx-
>p_norP.out.datafmt_d_tax,CS_REAL_TYPE,CS_REAL,2);
   DTAFMT_BND(stx-
>p_norP.out.datafmt_o_id,CS_INT_TYPE,CS_INT,3);
   DTAFMT_CHAR_BND(stx-
>p_norP.out.datafmt_c_last,CS_CHAR_TYPE,4,17);
   DTAFMT_BND(stx-
>p_norP.out.datafmt_c_discount,CS_REAL_TYPE,CS_REAL
,5);
   DTAFMT_CHAR_BND(stx-
>p_norP.out.datafmt_c_credit,CS_CHAR_TYPE,6,3);
   DTAFMT_BND(stx-
>p_norP.out.datafmt_o_entry_d,CS_DATETIME_TYPE,CS_
DATETIME,7);
/*
   RTN_ON_FAIL(stx->context,
      ct_cmd_props(stx->nor_local_cmd, CS_SET,
CS_STICKY_BINDS, &stx->sticky_binds, CS_UNUSED,
NULL),
    "init_nord: ct_cmd_props() failed.");

   RTN_ON_FAIL(stx->context,
      ct_cmd_props(stx->nor_remote_cmd, CS_SET,
CS_STICKY_BINDS, &stx->sticky_binds, CS_UNUSED,
NULL),
    "init_nord: ct_cmd_props() failed.");
*/
   CT_CMD(stx->nor_local_cmd,CS_RPC_CMD,
"neworder_local");
   CT_CMD(stx->nor_remote_cmd,CS_RPC_CMD,
"neworder_remote");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->nor_local_cmd, &stx-
>p_norP.in.datafmt_w_id,&stx->norP.in_nord.s_W_ID, &stx-
>norP.in_nord.w_id_len, &stx->nonullind),
      "[NO] ct_param() for @param1 failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->nor_local_cmd, &stx-
>p_norP.in.datafmt_d_id,&stx->norP.in_nord.s_D_ID, &stx-
>norP.in_nord.d_id_len, &stx->nonullind),
      "[NO] ct_param() for @param2 failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->nor_local_cmd, &stx-
>p_norP.in.datafmt_c_id,&stx->norP.in_nord.s_C_ID, &stx-
>norP.in_nord.c_id_len, &stx->nonullind),
      "[NO] ct_param() for @param3 failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->nor_local_cmd, &stx-
>p_norP.in.datafmt_ol_cnt,&stx->norP.in_nord.s_O_OL_CNT,
&stx->norP.in_nord.ol_cnt_len, &stx->nonullind),
      "[NO] ct_param() for @param4 failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->nor_remote_cmd, &stx-
>p_norP.in.datafmt_w_id,&stx->norP.in_nord.s_W_ID, &stx-
>norP.in_nord.w_id_len, &stx->nonullind),
      "[NO] ct_param() for @param1 failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->nor_remote_cmd, &stx-
>p_norP.in.datafmt_d_id,&stx->norP.in_nord.s_D_ID, &stx-
>norP.in_nord.d_id_len, &stx->nonullind),
      "[NO] ct_param() for @param2 failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->nor_remote_cmd, &stx-
>p_norP.in.datafmt_c_id,&stx->norP.in_nord.s_C_ID, &stx-
>norP.in_nord.c_id_len, &stx->nonullind),
      "[NO] ct_param() for @param3 failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->nor_remote_cmd, &stx-
>p_norP.in.datafmt_ol_cnt,&stx->norP.in_nord.s_O_OL_CNT,
&stx->norP.in_nord.ol_cnt_len, &stx->nonullind),
      "[NO] ct_param() for @param4 failed.");
```

```
  for (i=0;i<ITEM_LIST;i++)
  {
     RTN_ON_FAIL(stx->context,
        ct_setparam(stx->nor_local_cmd, &stx-
>p_norP.in.ol[i].datafmt_i_id, &(stx-
>norP.in_nord.s_OL_I_ID[i]), &stx->norP.in_nord.i_id_len,
&stx->nonullind),
        "[NO] ct_setparam() for @i_id failed.");
     RTN_ON_FAIL(stx->context,
        ct_setparam(stx->nor_local_cmd, &stx-
>p_norP.in.ol[i].datafmt_qty, &(stx-
>norP.in_nord.s_OL_QUANTITY[i]), &stx-
>norP.in_nord.qty_len, &stx->nonullind),
        "[NO] ct_setparam() for @qnt failed.");
  }

  for (i=0;i<ITEM_LIST;i++)
  {
     RTN_ON_FAIL(stx->context,
        ct_setparam(stx->nor_remote_cmd, &stx-
>p_norP.in.ol[i].datafmt_i_id, &(stx-
>norP.in_nord.s_OL_I_ID[i]), &stx->norP.in_nord.i_id_len,
&stx->nonullind),
        "[NO] ct_setparam() for @i_id failed.");
     RTN_ON_FAIL(stx->context,
        ct_setparam(stx->nor_remote_cmd, &stx-
>p_norP.in.ol[i].datafmt_supply_w_id, &(stx-
>norP.in_nord.s_OL_SUPPLY_W_ID[i]), &stx-
>norP.in_nord.s_w_id_len, &stx->nonullind),
        "[NO] ct_setparam() for @s_w_id failed.");
     RTN_ON_FAIL(stx->context,
        ct_setparam(stx->nor_remote_cmd, &stx-
>p_norP.in.ol[i].datafmt_qty, &(stx-
>norP.in_nord.s_OL_QUANTITY[i]), &stx-
>norP.in_nord.qty_len, &stx->nonullind),
        "[NO] ct_setparam() for @qnt failed.");
  }

  ERRORMSG("<<init_nord"<<endl);
}

extern "C" void init_stok(void *ctx)
{
  syb_con_data_t    *stx = (syb_con_data_t *)ctx;
  CS_INT         rc;

  ERRORMSG(">>init_stok"<<endl);
  //allocate a new command structure

  ERRORMSG("allocated cmd"<<endl);

  //allocate data fields
  DTAFMT(stx-
>p_stoP.in.datafmt_w_id,CS_SMALLINT_TYPE,"@w_id");
  DTAFMT(stx-
>p_stoP.in.datafmt_d_id,CS_TINYINT_TYPE,"@d_id");
  DTAFMT(stx-
>p_stoP.in.datafmt_threshold,CS_SMALLINT_TYPE,"@thres
hold");

  DTAFMT_BND(stx-
>p_stoP.out.datafmt_low_count,CS_INT_TYPE,CS_INT,1);

  RTN_ON_FAIL(stx->context,
     ct_cmd_alloc(stx->conn, &stx->sto_cmd),
  "Could not allocate StockLevel command structure.");

  RTN_ON_FAIL(stx->context,
     ct_cmd_props(stx->sto_cmd, CS_SET,
CS_STICKY_BINDS, &stx->sticky_binds, CS_UNUSED,
NULL),
   "init_stok: ct_cmd_props() failed.");

  //Init rpc
  CT_CMD(stx->sto_cmd,CS_RPC_CMD, "stock_level");

  ERRORMSG("initialized stock_level stored
procedure"<<endl);
  RTN_ON_FAIL(stx->context,
     ct_setparam(stx->sto_cmd, &stx-
>p_stoP.in.datafmt_w_id,&stx->stoP.in_stok.s_W_ID, &stx-
>stoP.in_stok.w_id_len, &stx->nonullind),
  "ct_setparam() for @param1 failed.");

  RTN_ON_FAIL(stx->context,
     ct_setparam(stx->sto_cmd, &stx-
>p_stoP.in.datafmt_d_id,&stx->stoP.in_stok.s_D_ID, &stx-
>stoP.in_stok.d_id_len, &stx->nonullind),
  "ct_setparam() for @param2 failed.");

  RTN_ON_FAIL(stx->context,
     ct_setparam(stx->sto_cmd, &stx-
>p_stoP.in.datafmt_threshold, &(stx-
>stoP.in_stok.s_threshold), &stx->stoP.in_stok.threshold_len,
&stx->nonullind),
  "ct_setparam() for @param3 failed.");
  ERRORMSG("<<init_stok"<<endl);
}

extern "C" void init_paym(void *ctx)
{
  syb_con_data_t    *stx = (syb_con_data_t *)ctx;
  CS_INT         rc;

  ERRORMSG(">>init_paym"<<endl);
  //allocate data fields
  DTAFMT(stx-
>p_payP.in_by_id.datafmt_w_id,CS_SMALLINT_TYPE,"@w
_id");
  DTAFMT(stx-
>p_payP.in_by_id.datafmt_c_w_id,CS_SMALLINT_TYPE,"@
c_w_id");
  DTAFMT(stx-
>p_payP.in_by_id.datafmt_h_amount_cents,CS_FLOAT_TY
PE,"@h_amount");
  DTAFMT(stx-
>p_payP.in_by_id.datafmt_d_id,CS_TINYINT_TYPE,"@d_id"
);
  DTAFMT(stx-
>p_payP.in_by_id.datafmt_c_d_id,CS_TINYINT_TYPE,"@c_
d_id");
  DTAFMT(stx-
>p_payP.in_by_id.datafmt_c_id,CS_INT_TYPE,"@c_id");

  DTAFMT(stx-
>p_payP.in_by_name.datafmt_w_id,CS_SMALLINT_TYPE,"
@w_id");
  DTAFMT(stx-
>p_payP.in_by_name.datafmt_c_w_id,CS_SMALLINT_TYPE
,"@c_w_id");
  DTAFMT(stx-
>p_payP.in_by_name.datafmt_h_amount_cents,CS_FLOAT_
TYPE,"@h_amount");
  DTAFMT(stx-
>p_payP.in_by_name.datafmt_d_id,CS_TINYINT_TYPE,"@d
_id");
  DTAFMT(stx-
>p_payP.in_by_name.datafmt_c_d_id,CS_TINYINT_TYPE,"
@c_d_id");
  DTAFMT_CHAR(stx-
>p_payP.in_by_name.datafmt_c_last,CS_CHAR_TYPE,"@c
_last",17);

  DTAFMT_BND(stx-
>p_payP.out.datafmt_c_id,CS_INT_TYPE,CS_INT,1);
  DTAFMT_CHAR_BND(stx-
>p_payP.out.datafmt_c_last,CS_CHAR_TYPE,2,17);
  DTAFMT_BND(stx-
>p_payP.out.datafmt_h_date,CS_DATETIME_TYPE,CS_DA
TETIME,3);
  DTAFMT_CHAR_BND(stx-
>p_payP.out.datafmt_w_street_1,CS_CHAR_TYPE,4,21);
  DTAFMT_CHAR_BND(stx-
>p_payP.out.datafmt_w_street_2,CS_CHAR_TYPE,5,21);
```

```
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_w_city,CS_CHAR_TYPE,6,21);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_w_state,CS_CHAR_TYPE,7,3);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_w_zip,CS_CHAR_TYPE,8,10);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_d_street_1,CS_CHAR_TYPE,9,21);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_d_street_2,CS_CHAR_TYPE,10,21);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_d_city,CS_CHAR_TYPE,11,21);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_d_state,CS_CHAR_TYPE,12,3);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_d_zip,CS_CHAR_TYPE,13,10);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_first,CS_CHAR_TYPE,14,17);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_middle,CS_CHAR_TYPE,15,3);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_street_1,CS_CHAR_TYPE,16,21);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_street_2,CS_CHAR_TYPE,17,21);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_city,CS_CHAR_TYPE,18,21);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_state,CS_CHAR_TYPE,19,3);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_zip,CS_CHAR_TYPE,20,10);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_phone,CS_CHAR_TYPE,21,17);
   DTAFMT_BND(stx->p_payP.out.datafmt_c_since,CS_DATETIME_TYPE,CS_DATETIME,22);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_credit,CS_CHAR_TYPE,23,3);
   DTAFMT_BND(stx->p_payP.out.datafmt_c_credit_lim,CS_FLOAT_TYPE,CS_FLOAT,24);
   DTAFMT_BND(stx->p_payP.out.datafmt_c_discount,CS_REAL_TYPE,CS_REAL,25);
   DTAFMT_BND(stx->p_payP.out.datafmt_c_balance,CS_FLOAT_TYPE,CS_FLOAT,26);
   DTAFMT_CHAR_BND(stx->p_payP.out.datafmt_c_data,CS_CHAR_TYPE,27,201);

   RTN_ON_FAIL(stx->context,
      ct_cmd_alloc(stx->conn, &stx->pay_id_cmd),
      "Could not allocate command structure.");

   RTN_ON_FAIL(stx->context,
      ct_cmd_alloc(stx->conn, &stx->pay_name_cmd),
      "Could not allocate command structure.");

   RTN_ON_FAIL(stx->context,
      ct_cmd_props(stx->pay_name_cmd, CS_SET,
CS_STICKY_BINDS, &stx->sticky_binds, CS_UNUSED,
NULL),
      "init_dlvy: ct_cmd_props() failed.");

   RTN_ON_FAIL(stx->context,
      ct_cmd_props(stx->pay_id_cmd, CS_SET,
CS_STICKY_BINDS, &stx->sticky_binds, CS_UNUSED,
NULL),
      "init_dlvy: ct_cmd_props() failed.");

   CT_CMD(stx->pay_name_cmd,CS_RPC_CMD,
"payment_byname");
   CT_CMD(stx->pay_id_cmd,CS_RPC_CMD,
"payment_byid");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_name_cmd, &stx->p_payP.in_by_name.datafmt_w_id,&stx->payP.in_paym.s_W_ID, &stx->payP.in_paym.w_id_len,
&stx->nonullind),
      "ct_param() for @w_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_name_cmd, &stx->p_payP.in_by_name.datafmt_c_w_id,&stx->payP.in_paym.s_C_W_ID,  &stx->payP.in_paym.c_w_id_len,
&stx->nonullind),
      "ct_param() for @c_w_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_name_cmd, &stx->p_payP.in_by_name.datafmt_h_amount_cents,&stx->payP.in_paym.s_H_AMOUNT, &stx->payP.in_paym.h_amount_cents_len, &stx->nonullind),
      "ct_param() for @h_amount failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_name_cmd, &stx->p_payP.in_by_name.datafmt_d_id,&stx->payP.in_paym.s_D_ID, &stx->payP.in_paym.d_id_len, &stx->nonullind),
      "ct_param() for @d_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_name_cmd, &stx->p_payP.in_by_name.datafmt_c_d_id,&stx->payP.in_paym.s_C_D_ID, &stx->payP.in_paym.c_d_id_len,
&stx->nonullind),
      "ct_param() for @c_d_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_name_cmd, &stx->p_payP.in_by_name.datafmt_c_last,&stx->payP.in_paym.s_C_LAST, &stx->payP.in_paym.c_last_len,
&stx->nonullind),
      "ct_param() for @c_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_id_cmd, &stx->p_payP.in_by_id.datafmt_w_id,&stx->payP.in_paym.s_W_ID, &stx->payP.in_paym.w_id_len,
&stx->nonullind),
      "ct_param() for @w_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_id_cmd, &stx->p_payP.in_by_id.datafmt_c_w_id,&stx->payP.in_paym.s_C_W_ID,  &stx->payP.in_paym.c_w_id_len,
&stx->nonullind),
      "ct_param() for @c_w_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_id_cmd, &stx->p_payP.in_by_id.datafmt_h_amount_cents,&stx->payP.in_paym.s_H_AMOUNT, &stx->payP.in_paym.h_amount_cents_len, &stx->nonullind),
      "ct_param() for @h_amount failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_id_cmd, &stx->p_payP.in_by_id.datafmt_d_id,&stx->payP.in_paym.s_D_ID, &stx->payP.in_paym.d_id_len, &stx->nonullind),
      "ct_param() for @d_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_id_cmd, &stx->p_payP.in_by_id.datafmt_c_d_id,&stx->payP.in_paym.s_C_D_ID, &stx->payP.in_paym.c_d_id_len,
&stx->nonullind),
      "ct_param() for @c_d_id failed.");

   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->pay_id_cmd, &stx->p_payP.in_by_id.datafmt_c_id,&stx->payP.in_paym.s_C_ID, &stx->payP.in_paym.c_id_len, &stx->nonullind),
      "ct_param() for @c_id failed.");
```

```
   ERRORMSG("<<init_paym"<<endl);
}

extern "C" void init_ords(void *ctx)
{
   syb_con_data_t    *stx = (syb_con_data_t *)ctx;
   CS_INT         rc;

   //allocate data fields
   ERRORMSG(">>init_ords"<<endl);
   DTAFMT_TST(stx-
>p_ordP.in_by_id.datafmt_w_id,CS_SMALLINT_TYPE,"@w_
id",sizeof(CS_SMALLINT));
   DTAFMT_TST(stx-
>p_ordP.in_by_id.datafmt_d_id,CS_TINYINT_TYPE,"@d_id"
,sizeof(CS_TINYINT));
   DTAFMT_TST(stx-
>p_ordP.in_by_id.datafmt_c_id,CS_INT_TYPE,"@c_id",sizeo
f(CS_INT));

   DTAFMT_TST(stx-
>p_ordP.in_by_name.datafmt_w_id,CS_SMALLINT_TYPE,"
@w_id",sizeof(CS_SMALLINT));
   DTAFMT_TST(stx-
>p_ordP.in_by_name.datafmt_d_id,CS_TINYINT_TYPE,"@d
_id",sizeof(CS_TINYINT));
   DTAFMT_CHAR(stx-
>p_ordP.in_by_name.datafmt_c_last,CS_CHAR_TYPE,"@c_
last",17);

   DTAFMT_BND(stx-
>p_ordP.out.datafmt_ol_supply_w_id,CS_INT_TYPE,CS_IN
T,1);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_ol_i_id,CS_INT_TYPE,CS_INT,2);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_ol_quantity,CS_INT_TYPE,CS_INT,3);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_ol_amount,CS_FLOAT_TYPE,CS_FLO
AT,4);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_ol_delivery_d,CS_DATETIME_TYPE,C
S_DATETIME,5);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_c_id,CS_INT_TYPE,CS_INT,1);
   DTAFMT_CHAR_BND(stx-
>p_ordP.out.datafmt_c_last,CS_CHAR_TYPE,2,17);
   DTAFMT_CHAR_BND(stx-
>p_ordP.out.datafmt_c_first,CS_CHAR_TYPE,3,17);
   DTAFMT_CHAR_BND(stx-
>p_ordP.out.datafmt_c_middle,CS_CHAR_TYPE,4,3);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_c_balance,CS_FLOAT_TYPE,CS_FLO
AT,5);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_o_id,CS_INT_TYPE,CS_INT,6);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_o_entry_d,CS_DATETIME_TYPE,CS_
DATETIME,7);
   DTAFMT_BND(stx-
>p_ordP.out.datafmt_o_carrier_id,CS_INT_TYPE,CS_INT,8);

   RTN_ON_FAIL(stx->context,
   ct_cmd_alloc(stx->conn, &stx->ord_name_cmd),
   "Could not allocate command structure.");

   RTN_ON_FAIL(stx->context,
      ct_cmd_alloc(stx->conn, &stx->ord_id_cmd),
   "Could not allocate command structure.");

   RTN_ON_FAIL(stx->context,
      ct_cmd_props(stx->ord_name_cmd, CS_SET,
CS_STICKY_BINDS, &stx->sticky_binds, CS_UNUSED,
NULL),
    "init_ords: ct_cmd_props() failed.");
   RTN_ON_FAIL(stx->context,
      ct_cmd_props(stx->ord_id_cmd, CS_SET,
CS_STICKY_BINDS, &stx->sticky_binds, CS_UNUSED,
NULL),
    "init_ords: ct_cmd_props() failed.");

   CT_CMD(stx->ord_name_cmd,CS_RPC_CMD,
"order_status_byname");
   CT_CMD(stx->ord_id_cmd,CS_RPC_CMD,
"order_status_byid");



   ERRORMSG(">>Allocating param1"<<endl);
   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->ord_name_cmd, &stx-
>p_ordP.in_by_name.datafmt_w_id,(CS_VOID *)&stx-
>ordP.in_ords.s_W_ID, &stx->ordP.in_ords.w_id_len, &stx-
>nonullind),
      "[OrderStatus] ct_param() for @param1 failed.");
   ERRORMSG(">>Allocating param2"<<endl);
   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->ord_name_cmd, &stx-
>p_ordP.in_by_name.datafmt_d_id,(CS_VOID *)&stx-
>ordP.in_ords.s_D_ID, &stx->ordP.in_ords.d_id_len, &stx-
>nonullind),
      "[OrderStatus] ct_param() for @param2 failed.");
   ERRORMSG(">>Allocating param3"<<endl);
   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->ord_name_cmd, &stx-
>p_ordP.in_by_name.datafmt_c_last, (CS_VOID *)&stx-
>ordP.in_ords.s_C_LAST, &stx->ordP.in_ords.c_last_len,
&stx->nonullind),
      "[OrderStatus] ct_param() for @param3 failed.");

   RTN_ON_FAIL(stx->context, rc, "Could not initiate
order_status_byid RPC command.");
   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->ord_id_cmd, &stx-
>p_ordP.in_by_id.datafmt_w_id,&stx->ordP.in_ords.s_W_ID,
&stx->ordP.in_ords.w_id_len, &stx->nonullind),
      "[OrderStatus] ct_param() for @param1 failed.");
   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->ord_id_cmd, &stx-
>p_ordP.in_by_id.datafmt_d_id,&stx->ordP.in_ords.s_D_ID,
&stx->ordP.in_ords.d_id_len, &stx->nonullind),
      "[OrderStatus] ct_param() for @param2 failed.");
   RTN_ON_FAIL(stx->context,
      ct_setparam(stx->ord_id_cmd, &stx-
>p_ordP.in_by_id.datafmt_c_id, &(stx->ordP.in_ords.s_C_ID),
&stx->ordP.in_ords.c_id_len, &stx->nonullind),
      "[OrderStatus] ct_param() for @param3 failed.");

   ERRORMSG("<<init_ords"<<endl);

}

/*
**********************************************************************
** Name         :  connect_db
** Description :
**            Function calls sybase api to connect to db
** Parameters:
**            char*    dbName
**            void**   uninitialized context
** Returns      :
**            int - return code
** Comments     :
**            To connect to db, first connection must be
**            established.  Next, context for that connect
**            be saved off.  Finally, detach from the
**            context just created.
**
**********************************************************************
*/
extern "C" TPCCSYBASEGLUE_API int connect_db(char
*dbName,void **ctx)
{
   syb_con_data_t    *stx = (syb_con_data_t *)ctx;
   CS_INT         ret=0;
   CS_INT         *outlen = NULL;
```

```
   CS_INT val;
HKEY   registryKey;
DWORD  regType;
char   value[MAX_STRING_LEN];
char   msgstr[4096] = {NULL};
DWORD  regValueSize = MAX_STRING_LEN;


   *ctx = conHandle = malloc(sizeof(syb_con_data_t));


   stx = (syb_con_data_t *)*ctx;

   EnterCriticalSection(&errorMutex);
   stx->tid = count++;
   ERRORMSG("Processing thread "<<stx->tid<<endl);
   LeaveCriticalSection(&errorMutex);

   stx->sticky_binds = CS_TRUE;
   stx->nonullind = 0;
   stx->nullind = -1;

   if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTERY
_SUB_KEY,0,KEY_READ,&registryKey) ==
ERROR_SUCCESS)
   {
      DEBUGMSG("Registry key open"<<endl);
      //get the null db user name
      regValueSize = sizeof(value);
      if
(RegQueryValueEx(registryKey,ENGINE_APP_NAME,0,&reg
Type,(BYTE *) &value,&regValueSize)==
ERROR_SUCCESS )
      {
         strncpy(stx->engine_app,value,regValueSize);
         stx->engine_app[regValueSize] = '\0';
      }
      else
         return ERR_INVALID_APPNAME;
      ERRORMSG("app engine name:"<< stx->engine_app <<
" len "<< strlen(stx->engine_app)<< endl);
      RegCloseKey(registryKey);
      DEBUGMSG("app engine name:"<< stx->engine_app <<
endl);
   } else {
      ERRORMSG("Cannot open registry");
      DEBUGMSG("Cannot open registry");
      return -1;
   }
```

```
   DEBUGMSG("Entered sybaseGlue do_connect using
dbName:"<< dbName << endl << "setting username:"<<
userName << " password:" <<userPassword << endl);
   //allocate sybase CS_CONTEXT structure
   DEBUGMSG("Allocating CS_CONTEXT"<<endl);
   if (stx->tid == 0) {
      ERRORMSG("cs_ctx_alloc on thread "<<stx->tid<<endl);
      global_context = (CS_CONTEXT *)NULL;
      ret = cs_ctx_alloc(CS_VERSION_100, &global_context);
      if (ret != CS_SUCCEED)
      {
         ERRORMSG("cs_ctx_alloc failed"<<endl);
      }
      /*
      ** Initialize Client-Library.
      */
      ret = ct_init(global_context, CS_VERSION_100);
      if (ret != CS_SUCCEED) {
         ERRORMSG("ct_init failed thread "<<stx->tid<<endl);
      }

      /*
      ** Step 2: Set up the error handling. Install callback
handlers
      ** for: - CS-Library errors - Client-Library errors - Server
      ** messages.
      */

      /*
      ** Install a callback function to handle CS-Library errors.
      */
      ret = cs_config(global_context, CS_SET,
CS_MESSAGE_CB,
         (CS_VOID *)clientmsg_cb,
         CS_UNUSED, NULL);
      if (ret != CS_SUCCEED) {
         ERRORMSG("cs_config(CS_MESSAGE_CB)
failed"<<endl);
      }

      /*
      ** Install a callback function to handle Client-Library
errors.
      **
      ** The client message callback receives error or
informational
      ** messages discovered by Client-Library.
      */
      ret = ct_callback(global_context, NULL, CS_SET,
CS_CLIENTMSG_CB,
         (CS_VOID *)clientmsg_cb);
      if (ret != CS_SUCCEED)
      {
         ERRORMSG("ct_callback for client messages
failed"<<endl);
      }

      /*
      ** The server message callback receives server
messages sent by
      ** the server. These are error or inforamational
messages.
      */
      ret = ct_callback(global_context, NULL, CS_SET,
CS_SERVERMSG_CB,
         (CS_VOID *)servermsg_cb);
      if (ret != CS_SUCCEED)
      {
         ERRORMSG("ct_callback for server messages
failed"<<endl);
      }
       int maxConnects = 50;
       ret = ct_config (global_context, CS_SET,
CS_MAX_CONNECT,
                  &maxConnects, CS_UNUSED, NULL  );
      if (ret != CS_SUCCEED)
      {

          ERRORMSG ( "ct_config failed to set
CS_MAX_CONNECT to 50");
      }
      ERRORMSG("Releasing semaphore thread "<<stx-
>tid<<endl);
      ReleaseSemaphore(ASemaphore, 100, NULL);
   } else {
      ERRORMSG("Waiting for semaphore thread "<<stx-
>tid<<endl);
      WaitForSingleObject(ASemaphore, INFINITE);
      ERRORMSG("Received semaphore thread "<<stx-
>tid<<endl);
   }
   stx->context = global_context;

   /*
   ** Step 3: Connect to the server. We must: - Allocate a
connection
   ** structure. - Set user name and password. - Create the
   ** connection.
   */

   /*
   ** First, allocate a connection structure.
   */
   ret = ct_con_alloc(global_context, &stx->conn);
```

```
  if (ret != CS_SUCCEED)
  {
     ERRORMSG("ct_con_alloc() failed"<<endl);
  }

  /*
  ** These two calls set the user credentials (username and
  ** password) for opening the connection.
  */
  ret = ct_con_props(stx->conn, CS_SET, CS_USERNAME,
          "sa", CS_NULLTERM, NULL);
  if (ret != CS_SUCCEED)
  {
     ERRORMSG("Could not set user name"<<endl);
  }
  ret = ct_con_props(stx->conn, CS_SET, CS_PASSWORD,
          "", CS_NULLTERM, NULL);
  if (ret != CS_SUCCEED)
  {
     ERRORMSG("Could not set password"<<endl);
  }
  val = 4096;
 ret = ct_con_props(stx->conn, CS_SET, CS_PACKETSIZE,
          &val, CS_UNUSED, NULL);
  if (ret != CS_SUCCEED)
  {
     ERRORMSG("Could not set password"<<endl);
  }

  ret = ct_con_props(stx->conn, CS_SET, CS_APPNAME,
          stx->engine_app, strlen(stx->engine_app), NULL);
  if (ret != CS_SUCCEED)
  {
     ERRORMSG("Could not set app"<<endl);
  }

   int mytds_version = CS_TDS_46;
  ret = ct_con_props(stx->conn, CS_SET,
CS_TDS_VERSION, &mytds_version,
      CS_UNUSED, NULL);
  if (ret != CS_SUCCEED)
  {
     ERRORMSG("Could not set app"<<endl);
  }


  /*
  ** Create the connection.
  */
  ret = ct_connect(stx->conn, (CS_CHAR *)NULL, 0);
  if (ret != CS_SUCCEED)
  {
     ERRORMSG("Could not connect!"<<endl);
  }

  if (ret != CS_SUCCEED)
  {
     CS_INTmsgno = 0;
     CS_CHAR clientmsg[4096]={NULL};

     ret = ct_diag  (stx->conn, CS_GET,
CS_CLIENTMSG_TYPE ,msgno, &clientmsg);

  DEBUGMSG("numofmessages:"<<msgno<<endl<<"Messa
ges:"<<clientmsg<<endl);

  ERRORMSG("numofmessages:"<<msgno<<endl<<"Messa
ges:"<<clientmsg<<endl);
     //get_client_msgs(stx);

     ERRORMSG("Connect to db failed, ret:"<<ret<<endl);
     DEBUGMSG("Connect to db failed, ret:"<<ret<<endl);

     return -1;
  }
  ERRORMSG("Connected to db"<<endl);
  DEBUGMSG("Connected to db"<<endl);

  if ((ret = ex_use_db(stx->conn, "tpcc")) != CS_SUCCEED)
  {
     ERRORMSG(" ex_use_db(\"tpcc\") failed");
     return -1;
  }
/*
  if ((ret = ct_debug(stx->context, stx->conn,
CS_SET_DBG_FILE,  CS_UNUSED,
"c:\\InetPub\\wwwroot\\tpcc\\debug.log", CS_NULLTERM)) !=
CS_SUCCEED)
  {
     ERRORMSG(" ct_debug() failed");
     return -1;
  }

  if ((ret = ct_debug(stx->context, stx->conn, CS_SET_FLAG,
CS_DBG_ALL, NULL, CS_UNUSED)) != CS_SUCCEED)
  {
     ERRORMSG(" ct_debug() failed");
     return -1;
  }
*/
  DEBUGMSG("Using tpcc db"<<endl);
  init_nord(stx);
  init_paym(stx);
  init_ords(stx);
  init_stok(stx);
  init_dlvy(stx);
  init_rtn(stx);

  return OK;
}

/*
****************************************************************
** Name        :  DBInit
** Description :
**            Function to initialize benchmark env
** Parameters:
**            void*     stored context
** Returns     :
**            int - return code
** Comments    :
**            To disconnect from db, first must attach to
**            thread's context.  Next, disconnect from db
****************************************************************
*/

extern "C" TPCCSYBASEGLUE_API int DBInit(void *ctx)
{
  syb_con_data_t    *stx = (syb_con_data_t *)ctx;

  return OK;
}
/*
****************************************************************
** Name        :  disconnect_db
** Description :
**            Function calls db2 api to disconnect from db
** Parameters:
**            void*     stored context
** Returns     :
**            int - return code
** Comments    :
**            To disconnect from db, first must attach to
**            thread's context.  Next, disconnect from db
****************************************************************
*/

extern "C" TPCCSYBASEGLUE_API int disconnect_db(void
*ctx)
{

  syb_con_data_t    *stx = (syb_con_data_t *)ctx;
  CS_RETCODE    ret;

  ERRORMSG("disconnect_db called"<<endl);
  ret = ct_close(stx->conn, CS_UNUSED);
```

```
    EXIT_ON_FAIL(stx->context, ret, "Orderly connection-
close failed.");

  ret = ct_con_drop(stx->conn);
    EXIT_ON_FAIL(stx->context, ret, "ct_con_drop() failed.");

  ret = ct_cmd_drop(stx->nor_local_cmd);
   EXIT_ON_FAIL(stx->context, ret, "ct_cmd_drop() failed.");

  ret = ct_cmd_drop(stx->nor_remote_cmd);
   EXIT_ON_FAIL(stx->context, ret, "ct_cmd_drop() failed.");

  ret = ct_cmd_drop(stx->pay_id_cmd);
   EXIT_ON_FAIL(stx->context, ret, "ct_cmd_drop() failed.");

  ret = ct_cmd_drop(stx->pay_name_cmd);
   EXIT_ON_FAIL(stx->context, ret, "ct_cmd_drop() failed.");

  ret = ct_cmd_drop(stx->ord_id_cmd);
   EXIT_ON_FAIL(stx->context, ret, "ct_cmd_drop() failed.");

  ret = ct_cmd_drop(stx->ord_name_cmd);
   EXIT_ON_FAIL(stx->context, ret, "ct_cmd_drop() failed.");

  ret = ct_cmd_drop(stx->sto_cmd);
   EXIT_ON_FAIL(stx->context, ret, "ct_cmd_drop() failed.");

  ret = ct_cmd_drop(stx->del_cmd);
   EXIT_ON_FAIL(stx->context, ret, "ct_cmd_drop() failed.");

  ret = cs_ctx_drop(stx->context);
    EXIT_ON_FAIL(stx->context, ret, "ct_ctx_drop() failed.");

  free(ctx);
  ERRORMSG("disconnect_db finished"<<endl);
  return OK;
}

/*
*********************************************************************
** Name        :  do_nord
** Description :
**            Function calls db2 api to execute nord txn
** Parameters:
**            nord_wrapper*  new order txn structs wrapper
**            void*        stored context
** Returns     :
**            int - return code
** Comments    :
**            Attach to thread's context, call nord sql function
**            then detach from context.
*********************************************************************
*/
extern "C" TPCCSYBASEGLUE_API int
do_nord(nord_wrapper *nord,void *ctx)
{
  syb_con_data_t   *stx = (syb_con_data_t *)ctx;
  CS_COMMAND *cmd;
  CS_INT         rc,cols,rows,bound=0;
   int i;
  long result_type = 0;
  int terror = 0;
  int round = 1;
  int ordl=0;
  int deadlock = 0;
  CS_RETCODE    ret=CS_ROW_FAIL;

  DEBUGMSG("starting do_nord"<<endl);
  CopyMemory (&stx->norP,nord,sizeof(nord_wrapper));
  DEBUGMSG("nord in,"<<endl<<
     "nord c_id: " <<  nord->in_nord.s_C_ID << endl <<
     "nord w_id: " <<  nord->in_nord.s_W_ID << endl <<
     "nord d_id: " <<  nord->in_nord.s_D_ID << endl <<
     "nord ol_cnt: " <<  nord->in_nord.s_O_OL_CNT << endl);
  DEBUGMSG(">>Allocating cmd"<<endl);
  for (i=0;i<stx->norP.in_nord.s_O_OL_CNT;i++)
  {
     DEBUGMSG(
     "ordl i_id: " <<  nord->in_nord.s_OL_I_ID[i] << endl <<
     "ordl w_id: " <<  nord->in_nord.s_OL_SUPPLY_W_ID[i]
<< endl <<
     "ordl qty: " <<  nord->in_nord.s_OL_QUANTITY[i] <<
endl);
  }

  //allocate data fields

  stx->norP.in_nord.w_id_len = CS_SIZEOF(stx-
>norP.in_nord.s_W_ID);
  stx->norP.in_nord.d_id_len = CS_SIZEOF(stx-
>norP.in_nord.s_D_ID);
  stx->norP.in_nord.c_id_len = CS_SIZEOF(stx-
>norP.in_nord.s_C_ID);
  stx->norP.in_nord.ol_cnt_len = CS_SIZEOF(stx-
>norP.in_nord.s_O_OL_CNT);
  stx->norP.in_nord.i_id_len = CS_SIZEOF(stx-
>norP.in_nord.s_OL_I_ID[0]);
  stx->norP.in_nord.s_w_id_len = CS_SIZEOF(stx-
>norP.in_nord.s_OL_SUPPLY_W_ID[0]);
  stx->norP.in_nord.qty_len = CS_SIZEOF(stx-
>norP.in_nord.s_OL_QUANTITY[0]);

  nord->in_nord.s_all_local = 1;
  for (i=0;i<stx->norP.in_nord.s_O_OL_CNT;i++)
  {
    if (stx->norP.in_nord.s_OL_SUPPLY_W_ID[i] != stx-
>norP.in_nord.s_W_ID)
    {
      nord->in_nord.s_all_local = 0;
      break;
    }
  }

  if (nord->in_nord.s_all_local)
  {
    cmd = stx->nor_local_cmd;
  } else {
    cmd = stx->nor_remote_cmd;
  }
  sort_order_lines(&stx->norP.in_nord);

  do
  {
//    try
//    {
    deadlock = 0;
    JUST_RETURN( ct_send(cmd), "ct_send() failed.");

    DEBUGMSG("nord mid,"<<endl<<
         "nord c_id: " <<  nord->in_nord.s_C_ID << endl <<
         "nord w_id: " <<  nord->in_nord.s_W_ID << endl <<
         "nord d_id: " <<  nord->in_nord.s_D_ID << endl);
    DEBUGMSG("back from send"<<endl);

      /*
      ** Process the results.  Loop while ct_results() returns
CS_SUCCEED.
      */
    while ((rc = ct_results(cmd, &result_type)) ==
CS_SUCCEED)
    {
         //         if (deadlock)
         //         {
         //           ct_cancel((CS_CONNECTION *)NULL,
cmd, CS_CANCEL_ALL);
         //           ERRORMSG("ct_results: deadlock caught,
retrying."<<endl);
         //           ret = CS_END_DATA;
         //           break;
         //         }
      switch ((int)result_type)
      {
        case CS_END_RESULTS:
          DEBUGMSG("CS_END_RESULTS"<<endl);
          CS_INT val;
```

```
/*
            if (ex_fetch_data(cmd,&val) != CS_SUCCEED)
            {
               ERRORMSG("[NO]: ex_fetch_data()
failed"<<endl);
               return ret;
            }
*/
            if (do_rtn(ctx,cmd,&val) != CS_SUCCEED)
            {
               ERRORMSG("[NO]: do_rtn() failed"<<endl);
               return ret;
            }
            if (val == -3)
            {
               ERRORMSG("[NO]: do_rtn() returns -3,
indicating deadlock, retry..."<<endl);
               deadlock = 1;
            }
            ret = CS_END_DATA;
            break;
         case CS_CMD_SUCCEED:
            DEBUGMSG("[NO]CS_CMD_SUCCEED no rows
were returned?"<<endl);
            /*
            ** This means no rows were returned.
            */
            break;

         case CS_CMD_DONE:
            DEBUGMSG("[NO]CS_CMD_DONE we're done
with one result set."<<endl);
            /*
            ** This means we're done with one result set.
            */
            break;

         case CS_STATUS_RESULT:
            DEBUGMSG("[NO]CS_STATUS_RESULT
"<<endl);
            CS_INT v;
/*
            if (ex_fetch_data(cmd,&v) != CS_SUCCEED)
            {
               ERRORMSG("[do_nord]: ex_fetch_data()
failed"<<endl);
               return ret;
            }
*/
            if (do_rtn(ctx,cmd,&v) != CS_SUCCEED)
            {
               ERRORMSG("[do_nord]: do_rtn() failed"<<endl);
               return ret;
            }
            DEBUGMSG("[NO]v="<<v<<endl);
            if (v == -3)
            {
               ERRORMSG("[NO]: do_rtn() returns -3,
indicating deadlock, retry..."<<endl);
               deadlock = 1;
            }
            if (v == 1)
            {
               if (terror != INVALID_ITEM)
               {
                  terror = 1;
                  ERRORMSG("[NO]Transaction error, return
code="<<v);
               }
            }
            break;
         case CS_CMD_FAIL:
            DEBUGMSG("CS_CMD_FAIL"<<endl);
            /*
            ** This means that the server encountered an error
while
            ** processing our command.
            */
            ERRORMSG("[do_nord] ct_results() returned
CMD_FAIL"<<endl);
            terror = 1;
            break;

         case CS_ROW_RESULT:
            DEBUGMSG("[NO]CS_ROW_RESULT"<<endl);



            if ((rc = ct_res_info(cmd, CS_NUMDATA, &cols,
CS_UNUSED, NULL))
               != CS_SUCCEED)
            {
               ERRORMSG("[do_nord] ct_res_info failed,
rc="<<rc<<endl);
            }
            else
            {
               DEBUGMSG("[do_nord] round 1,
cols="<<cols<<endl);
            }

            if (cols == 4)
            {

               DEBUGMSG("START ORDL"<<endl);
               /*
               ct_describe(cmd,1,&stx-
>p_norP.out.ol.datafmt_i_name);
               ct_describe(cmd,2,&stx-
>p_norP.out.ol.datafmt_i_price);
               ct_describe(cmd,3,&stx-
>p_norP.out.ol.datafmt_s_quantity);
               ct_describe(cmd,4,&stx-
>p_norP.out.ol.datafmt_b_g);
               */
               CT_BND(cmd,1,stx-
>p_norP.out.ol.datafmt_i_name,
                  stx->norP.out_nord.s_I_NAME[ordl]);

               CT_BND(cmd,2,stx-
>p_norP.out.ol.datafmt_i_price,
                  stx->norP.out_nord.s_I_PRICE[ordl]);

               CT_BND(cmd,3,stx-
>p_norP.out.ol.datafmt_s_quantity,
                  stx->norP.out_nord.s_S_QUANTITY[ordl]);

               CT_BND(cmd,4,stx->p_norP.out.ol.datafmt_b_g,
                  stx->norP.out_nord.s_brand_generic[ordl]);
            }
            else
            {
               DEBUGMSG("START rest of data "<<endl);
               /*
               ct_describe(cmd,1,&stx-
>p_norP.out.datafmt_w_tax);
               ct_describe(cmd,2,&stx-
>p_norP.out.datafmt_d_tax);
               ct_describe(cmd,3,&stx-
>p_norP.out.datafmt_o_id);
               ct_describe(cmd,4,&stx-
>p_norP.out.datafmt_c_last);
               ct_describe(cmd,5,&stx-
>p_norP.out.datafmt_c_discount);
               ct_describe(cmd,6,&stx-
>p_norP.out.datafmt_c_credit);
               ct_describe(cmd,7,&stx-
>p_norP.out.datafmt_o_entry_d);
               */
               CT_BND(cmd,1,stx->p_norP.out.datafmt_w_tax,
                  stx->norP.out_nord.s_W_TAX);
               CT_BND(cmd,2,stx->p_norP.out.datafmt_d_tax,
                  stx->norP.out_nord.s_D_TAX);
               CT_BND(cmd,3,stx->p_norP.out.datafmt_o_id,
                  stx->norP.out_nord.s_O_ID);
               CT_BND(cmd,4,stx->p_norP.out.datafmt_c_last,
                  stx->norP.out_nord.s_C_LAST);
```

```
            CT_BND(cmd,5,stx->p_norP.out.datafmt_c_discount,
              stx->norP.out_nord.s_C_DISCOUNT);
            CT_BND(cmd,6,stx->p_norP.out.datafmt_c_credit,
              stx->norP.out_nord.s_C_CREDIT);
            CT_BND(cmd,7,stx->p_norP.out.datafmt_o_entry_d,
              stx->norP.out_nord.s_O_ENTRY_D_base);

            DEBUGMSG("END Round2"<<endl);
          }

          while ((ret = ct_fetch(cmd,CS_UNUSED,CS_UNUSED,CS_UNUSED,&rows)) ==
            CS_SUCCEED)
          {
            //            if (deadlock)
            //            {
            //              ct_cancel((CS_CONNECTION *)NULL, cmd, CS_CANCEL_ALL);
            //              ERRORMSG("ct_fetch: deadlock caught, retrying."<<endl);
            //              ret = CS_END_DATA;
            //              break;
            //            }
            if (cols == 4)
            {
              DEBUGMSG("CT_FETCH ordl:"<<ordl<<endl);
              if ((stx->norP.out_nord.s_I_NAME[ordl][0] == '\0') ||
                (stx->norP.out_nord.s_I_NAME[ordl][0] == ' '))
              {
                terror = INVALID_ITEM;
                DEBUGMSG("Forced Rollback"<<endl);
                //              ct_cancel((CS_CONNECTION *)NULL, cmd, CS_CANCEL_ALL);
                //              ret = CS_END_DATA;
                //              break;
              }
              stx->norP.out_nord.s_OL_AMOUNT[ordl] = stx->norP.in_nord.s_OL_QUANTITY[ordl] *
                stx->norP.out_nord.s_I_PRICE[ordl];
              stx->norP.out_nord.s_total_amount += stx->norP.out_nord.s_OL_AMOUNT[ordl];
              DEBUGMSG("END CT_FETCH round "<<round<<" ordl:"<<ordl<<endl<<
                "I_NAME:"<<stx->norP.out_nord.s_I_NAME[ordl]<<endl<<
                "I_PRICE:"<<stx->norP.out_nord.s_I_PRICE[ordl]<<endl<<
                "S_QUANTITY:"<<stx->norP.out_nord.s_S_QUANTITY[ordl]<<endl<<
                "BRAND_GENERIC:"<<stx->norP.out_nord.s_brand_generic[ordl]<<endl<<
                "OL_AMOUNT:"<<stx->norP.out_nord.s_OL_AMOUNT[ordl]<<endl);
              ordl++;
            }
            else
            {
              DEBUGMSG("CT_FETCH round:"<<round<<"ordl:"<<ordl<<endl);

              convert_dt(stx->context,
                &stx->norP.out_nord.s_O_ENTRY_D_base,
                (char *)&stx->norP.out_nord.s_O_ENTRY_D_time);

              DEBUGMSG("END round:"<<round<<
                "s_W_TAX:"<<stx->norP.out_nord.s_W_TAX<<endl<<
                "s_D_TAX:"<<stx->norP.out_nord.s_D_TAX<<endl<<
                "s_O_ID:"<<stx->norP.out_nord.s_O_ID<<endl<<
                "s_C_LAST:"<<stx->norP.out_nord.s_C_LAST<<endl<<
                "s_C_DISCOUNT:"<<stx->norP.out_nord.s_C_DISCOUNT<<endl<<
                "s_C_CREDIT:"<<stx->norP.out_nord.s_C_CREDIT<<endl);
              //DEBUGMSG(stx->norP.out_nord.s_O_ENTRY_D_time<<endl);

              DEBUGMSG("END CT_FETCH round:"<<round<<" ordl:"<<ordl<<endl);
            }
          }
          stx->norP.out_nord.s_O_OL_CNT = ordl;
          DEBUGMSG("ct_fetch, ret="<<ret<<endl);
          break;
        default :
          ERRORMSG("[NO]Unknown return code "<<result_type<<endl);
        }
      }
//    }
//    catch (runtime_error e)
//    {
//      ERRORMSG("Runtime error: "<<e.what()<<". Rollback and Retry."<<endl);
//      (CS_VOID)ct_cancel((CS_CONNECTION *)NULL, cmd, CS_CANCEL_ALL);
//      deadlock = 1;
//    }
      if (deadlock)
      {
        ERRORMSG("deadlock condition, retrying"<<endl);
      }
    } while(deadlock);

    if (ret != CS_END_DATA)
    {
//    (CS_VOID)ct_cmd_drop(cmd);
      (CS_VOID)ct_cancel((CS_CONNECTION *)NULL, cmd, CS_CANCEL_ALL);
      ERRORMSG("[NO] ret != CS_END_RESULTS, ret="<<ret<<", rc="<<rc<<endl);
      ERRORMSG(endl<<"[NewOrder]"<<endl<<
        "s_W_ID "<<stx->norP.in_nord.s_W_ID<<endl<<
        "s_D_ID "<<stx->norP.in_nord.s_D_ID<<endl<<
        "s_C_ID "<<stx->norP.in_nord.s_C_ID<<endl<<
        "s_O_OL_CNT "<<stx->norP.in_nord.s_O_OL_CNT<<endl);
      terror = 1;
    }
    /*
    if (ct_cmd_drop(stx->cmd) != CS_SUCCEED)
    {
      ERRORMSG("[ordl] Cannot free cmd structure"<<endl);
    }
    */

// if (terror == INVALID_ITEM)
// {
//   ERRORMSG("Rollback caught");
// }

    stx->norP.out_nord.s_transtatus = terror;

    DEBUGMSG("Finished do_nord"<<endl);
    CopyMemory (nord,&stx->norP,sizeof(nord_wrapper));
    DEBUGMSG("nord out,"<<endl<<
      "nord c_id: " <<  nord->in_nord.s_C_ID << endl <<
      "nord w_id: " <<  nord->in_nord.s_W_ID << endl <<
      "nord d_id: " <<  nord->in_nord.s_D_ID << endl);
    return OK;
}

/*
*****************************************************************
** Name         :  do_pymt
** Description :
```

```
**            Function calls db2 api to execute pymt txn
** Parameters:
**            paym_wrapper* payment txn structs wrapper
**            void*        stored context
** Returns      :
**            int - return code
** Comments     :
**            Attach to thread's context, call nord sql function
**            then detach from context.
*****************************************************************
*/

extern "C" TPCCSYBASEGLUE_API int
do_pymt(paym_wrapper *pymt,void *ctx)
{
   syb_con_data_t    *stx = (syb_con_data_t *)ctx;
   CS_COMMAND *cmd;
   CS_INT         rc;
   CS_RETCODE     ret=CS_ROW_FAIL;
   long result_type = 0;
   int terror = 0,num_cols = 0;
   CS_DATETIME s_H_DATE_base;
   CS_DATETIME s_C_SINCE_base;
// static CS_INT first_transaction=1;

   pymt->in_paym.bylastname = ((pymt->in_paym.s_C_ID ==
0) ? 1 : 0);
   CopyMemory (&stx->payP,pymt,sizeof(paym_wrapper));
   stx->payP.in_paym.w_id_len = CS_SIZEOF(stx-
>payP.in_paym.s_W_ID);
   stx->payP.in_paym.c_w_id_len = CS_SIZEOF(stx-
>payP.in_paym.s_C_W_ID);
   stx->payP.in_paym.h_amount_cents_len =
CS_SIZEOF(stx->payP.in_paym.s_H_AMOUNT);
   stx->payP.in_paym.d_id_len = CS_SIZEOF(stx-
>payP.in_paym.s_D_ID);
   stx->payP.in_paym.c_d_id_len = CS_SIZEOF(stx-
>payP.in_paym.s_C_D_ID);
   stx->payP.in_paym.c_id_len = CS_SIZEOF(stx-
>payP.in_paym.s_C_ID);
   stx->payP.in_paym.c_last_len = strlen(stx-
>payP.in_paym.s_C_LAST);


   DEBUGMSG(">>Allocating cmd"<<endl);

   DEBUGMSG("paym in,"<<endl<<
      "s_W_ID "<<stx->payP.in_paym.s_W_ID<<endl<<
      "s_C_W_ID "<<stx->payP.in_paym.s_C_W_ID<<endl<<
      "s_H_AMOUNT "<<stx-
>payP.in_paym.s_H_AMOUNT<<endl<<
      "s_D_ID "<<(int)stx->payP.in_paym.s_D_ID<<endl<<
      "s_C_D_ID "<<(int)stx-
>payP.in_paym.s_C_D_ID<<endl<<
      "s_C_ID "<<stx->payP.in_paym.s_C_ID<<endl<<
      "s_W_ID "<<stx->payP.in_paym.s_W_ID<<endl<<
      "s_C_LAST "<<stx->payP.in_paym.s_C_LAST<<endl<<
      "bylastname "<<stx->payP.in_paym.bylastname<<endl);


   //allocate data fields


   if (stx->payP.in_paym.bylastname)
   {
      cmd = stx->pay_name_cmd;
   } else
   {
      cmd = stx->pay_id_cmd;
   }
   DEBUGMSG("Sending cmd"<<endl);
   JUST_RETURN( ct_send(cmd), "[Payment] ct_send()
failed.");

   /*
   ** Process the results.  Loop while ct_results() returns
CS_SUCCEED.
   */
   DEBUGMSG("Results"<<endl);
   DEBUGMSG("pymt mid,"<<endl<<
      "pymt c_id: " <<  pymt->in_paym.s_C_ID << endl <<
      "pymt w_id: " <<  pymt->in_paym.s_W_ID << endl <<
      "pymt d_id: " <<  pymt->in_paym.s_D_ID << endl);
   while ((rc = ct_results(cmd, &result_type)) ==
CS_SUCCEED)
   {
      switch ((int)result_type)
      {
      case CS_END_RESULTS:
         DEBUGMSG("CS_END_RESULTS"<<endl);
         CS_INT val;
/*
        if (ex_fetch_data(stx->sto_cmd,&val) != CS_SUCCEED)
         {
            ERRORMSG("[Payment]: ex_fetch_data()
failed"<<endl);
            return ret;
         }
*/
         if (do_rtn(ctx,stx->sto_cmd,&val) != CS_SUCCEED)
         {
            ERRORMSG("[Payment]: do_rtn() failed"<<endl);
            return ret;
         }
         DEBUGMSG("[PY]CS_END_RESULTS,v="<<val);
         ret = CS_END_DATA;
         break;
      case CS_CMD_SUCCEED:
         DEBUGMSG("[PY]CS_CMD_SUCCEED no rows were
returned?");
         /*
         ** This means no rows were returned.
         */
         break;

      case CS_CMD_DONE:
         DEBUGMSG("[PY]CS_CMD_DONE done with one
result set");
         /*
         ** This means we're done with one result set.
         */
         break;

      case CS_STATUS_RESULT:
         CS_INT v;
/*
         if (ex_fetch_data(cmd,&v) != CS_SUCCEED)
         {
            ERRORMSG("[Payment]: ex_fetch_data()
failed"<<endl);
            terror = 1;
            ERRORMSG("[PY]Transaction error, return
code="<<v);
         }
*/
         if (do_rtn(ctx,cmd,&v) != CS_SUCCEED)
         {
            ERRORMSG("[Payment]: do_rtn() failed"<<endl);
            terror = 1;
         }
         if (v == 1)
         {
            ERRORMSG("[PY]Transaction error, return
code="<<v);
            terror = 1;
         }
         DEBUGMSG("[PY]CS_STATUS_RESULT,v="<<v);
         break;
      case CS_CMD_FAIL:
         /*
         ** This means that the server encountered an error
while
         ** processing our command.
         */
         ERRORMSG("[Payment] ct_results() returned
CMD_FAIL"<<endl);
```

```
        terror = 1;
        break;

      case CS_ROW_RESULT:
        DEBUGMSG("[PY]CS_ROW_RESULT"<<endl);
//      if (first_transaction)
//      {
          CT_BND(cmd,1,stx->p_payP.out.datafmt_c_id,
            stx->payP.out_paym.s_C_ID);
          CT_BND(cmd,2,stx->p_payP.out.datafmt_c_last,
            stx->payP.out_paym.s_C_LAST);
          CT_BND(cmd,3,stx->p_payP.out.datafmt_h_date,
            s_H_DATE_base);
          CT_BND(cmd,4,stx->p_payP.out.datafmt_w_street_1,
            stx->payP.out_paym.s_W_STREET_1);
          CT_BND(cmd,5,stx->p_payP.out.datafmt_w_street_2,
            stx->payP.out_paym.s_W_STREET_2);
          CT_BND(cmd,6,stx->p_payP.out.datafmt_w_city,
            stx->payP.out_paym.s_W_CITY);
          CT_BND(cmd,7,stx->p_payP.out.datafmt_w_state,
            stx->payP.out_paym.s_W_STATE);
          CT_BND(cmd,8,stx->p_payP.out.datafmt_w_zip,
            stx->payP.out_paym.s_W_ZIP);
          CT_BND(cmd,9,stx->p_payP.out.datafmt_d_street_1,
            stx->payP.out_paym.s_D_STREET_1);
          CT_BND(cmd,10,stx-
>p_payP.out.datafmt_d_street_2,
            stx->payP.out_paym.s_D_STREET_2);
          CT_BND(cmd,11,stx->p_payP.out.datafmt_d_city,
            stx->payP.out_paym.s_D_CITY);
          CT_BND(cmd,12,stx->p_payP.out.datafmt_d_state,
            stx->payP.out_paym.s_D_STATE);
          CT_BND(cmd,13,stx->p_payP.out.datafmt_d_zip,
            stx->payP.out_paym.s_D_ZIP);
          CT_BND(cmd,14,stx->p_payP.out.datafmt_c_first,
            stx->payP.out_paym.s_C_FIRST);
          CT_BND(cmd,15,stx->p_payP.out.datafmt_c_middle,
            stx->payP.out_paym.s_C_MIDDLE);
          CT_BND(cmd,16,stx-
>p_payP.out.datafmt_c_street_1,
            stx->payP.out_paym.s_C_STREET_1);
          CT_BND(cmd,17,stx-
>p_payP.out.datafmt_c_street_2,
            stx->payP.out_paym.s_C_STREET_2);
          CT_BND(cmd,18,stx->p_payP.out.datafmt_c_city,
            stx->payP.out_paym.s_C_CITY);
          CT_BND(cmd,19,stx->p_payP.out.datafmt_c_state,
            stx->payP.out_paym.s_C_STATE);
          CT_BND(cmd,20,stx->p_payP.out.datafmt_c_zip,
            stx->payP.out_paym.s_C_ZIP);
          CT_BND(cmd,21,stx->p_payP.out.datafmt_c_phone,
            stx->payP.out_paym.s_C_PHONE);
          CT_BND(cmd,22,stx->p_payP.out.datafmt_c_since,
            s_C_SINCE_base);
          CT_BND(cmd,23,stx->p_payP.out.datafmt_c_credit,
            stx->payP.out_paym.s_C_CREDIT);
          CT_BND(cmd,24,stx-
>p_payP.out.datafmt_c_credit_lim,
            stx->payP.out_paym.s_C_CREDIT_LIM);
          CT_BND(cmd,25,stx-
>p_payP.out.datafmt_c_discount,
            stx->payP.out_paym.s_C_DISCOUNT);
          CT_BND(cmd,26,stx-
>p_payP.out.datafmt_c_balance,
            stx->payP.out_paym.s_C_BALANCE);
          CT_BND(cmd,27,stx->p_payP.out.datafmt_c_data,
            stx->payP.out_paym.s_C_DATA);
//        first_transaction = 0;
//      }
        DEBUGMSG("[PY]Fetching"<<endl);
        while ((ret =
ct_fetch(cmd,CS_UNUSED,CS_UNUSED,CS_UNUSED,NU
LL)) ==
          CS_SUCCEED)
        {
          convert_dt(stx->context,
            &s_H_DATE_base,
            (char *)&stx->payP.out_paym.s_H_DATE_time);
          convert_dt(stx->context,
            &s_C_SINCE_base,
            (char *)&stx->payP.out_paym.c_since);
          stx->payP.out_paym.s_C_CREDIT_LIM /= 100.0;
          stx->payP.out_paym.s_C_BALANCE /= 100.0;
        }
      }
    }

    if (ret != CS_END_DATA)
    {
      (CS_VOID)ct_cancel((CS_CONNECTION *)NULL, cmd,
CS_CANCEL_ALL);
//    (CS_VOID)ct_cmd_drop(stx->cmd);
      ERRORMSG("[Payment] ret != CS_END_RESULTS,
ret="<<ret<<", rc="<<rc<<endl);
      terror = 1;
    }
/*
    if (ct_cmd_drop(cmd) != CS_SUCCEED)
    {
      ERRORMSG("[StockLevel] Cannot free cmd
structure"<<endl);
    }
*/
    stx->payP.out_paym.s_transtatus = terror == 0 ? 0 : -1;
    CopyMemory (pymt,&stx->payP,sizeof(paym_wrapper));
    DEBUGMSG("pymt out,"<<endl<<
      "pymt c_id: " <<  pymt->in_paym.s_C_ID << endl <<
      "pymt w_id: " <<  pymt->in_paym.s_W_ID << endl <<
      "pymt d_id: " <<  pymt->in_paym.s_D_ID << endl);
    return OK;
}

/*
****************************************************************
** Name        :  do_ords
** Description :
**            Function calls db2 api to execute ords txn
** Parameters:
**            ords_wrapper*   order status txn structs wrapper
**            void*        stored context
** Returns     :
**            int - return code
** Comments    :
**            Attach to thread's context, call nord sql function
**            then detach from context.
****************************************************************
*/
extern "C"  TPCCSYBASEGLUE_API int
do_ords(ords_wrapper *ords,void *ctx)
{
    syb_con_data_t    *stx = (syb_con_data_t *)ctx;
    CS_COMMAND *cmd;
    CS_INT ordl = 0;
    CS_INT         rc, second_round = 0,cols,rows,r;
    long result_type = 0;
    int terror = 0,num_cols = 0;
    CS_DATETIME s_O_ENTRY_D_base;
    CS_RETCODE     ret=CS_ROW_FAIL;

 ords->in_ords.bylastname = ((ords->in_ords.s_C_ID == 0) ?
1 : 0);
    CopyMemory (&stx->ordP.in_ords,&ords-
>in_ords,sizeof(in_ordstat_struct));
    stx->ordP.in_ords.w_id_len = CS_SIZEOF(stx-
>ordP.in_ords.s_W_ID);
    stx->ordP.in_ords.d_id_len = CS_SIZEOF(stx-
>ordP.in_ords.s_D_ID);
    stx->ordP.in_ords.c_last_len = strlen(stx-
>ordP.in_ords.s_C_LAST);
    stx->ordP.in_ords.c_id_len = CS_SIZEOF(stx-
>ordP.in_ords.s_C_ID);
    DEBUGMSG("ord in "<<stx->ordP.in_ords.s_W_ID<<" "<<
      stx->ordP.in_ords.s_D_ID<<" "<<
      stx->ordP.in_ords.s_C_ID<<endl);
    DEBUGMSG(">>do_ords"<<endl);
```

```
  if (ords->in_ords.bylastname)
  {
    cmd = stx->ord_name_cmd;
  } else
  {
    cmd = stx->ord_id_cmd;
  }

  //Init rpc



  DEBUGMSG("ord mid "<<stx->ordP.in_ords.s_W_ID<<"
"<<
    stx->ordP.in_ords.s_D_ID<<" "<<
    stx->ordP.in_ords.s_C_ID<<endl);

  DEBUGMSG("Sending cmd"<<endl);
  JUST_RETURN( ct_send(cmd), "[OrderStatus] ct_send()
failed.");

  /*
  ** Process the results.  Loop while ct_results() returns
CS_SUCCEED.
  */
  DEBUGMSG("Results"<<endl);
  while ((rc = ct_results(cmd, &result_type)) ==
CS_SUCCEED)
  {
    switch ((int)result_type)
    {
    case CS_END_RESULTS:
      DEBUGMSG("[OS]CS_END_RESULTS"<<endl);
      CS_INT val;
/*
      if (ex_fetch_data(cmd,&val) != CS_SUCCEED)
      {
        ERRORMSG("[OrderStatus]: ex_fetch_data()
failed"<<endl);
        return ret;
      }
*/
      if (do_rtn(ctx,cmd,&val) != CS_SUCCEED)
      {
        ERRORMSG("[OrderStatus]: do_rtn() failed"<<endl);
        return ret;
      }
      ret = CS_END_DATA;
      break;
    case CS_CMD_SUCCEED:
      DEBUGMSG("[OS]CS_CMD_SUCCEED"<<endl);
      /*
      ** This means no rows were returned.
      */
      break;

    case CS_CMD_DONE:
      DEBUGMSG("[OS]CS_CMD_DONE"<<endl);
      /*
      ** This means we're done with one result set.
      */
      break;

    case CS_STATUS_RESULT:
      CS_INT v;
/*
      if (ex_fetch_data(cmd,&v) != CS_SUCCEED)
      {
        ERRORMSG("[OrderStatus]: ex_fetch_data()
failed"<<endl);
        terror = 1;
      }
*/
      if (do_rtn(ctx,cmd,&v) != CS_SUCCEED)
      {
        ERRORMSG("[OrderStatus]: do_rtn() failed"<<endl);
        terror = 1;
      }
      if (v == 1)
      {
        ERRORMSG("[OS]Transaction error, return
code="<<v);
        terror = 1;
      }
      DEBUGMSG("[OS]CS_STATUS_RESULT,v="<<v);
      break;
    case CS_CMD_FAIL:
      /*
      ** This means that the server encountered an error
while
      ** processing our command.
      */
      ERRORMSG("[OrderStatus] ct_results() returned
CMD_FAIL"<<endl);
      terror = 1;
      break;

    case CS_ROW_RESULT:
      DEBUGMSG("[OS]CS_ROW_RESULT"<<endl);
      if (second_round)
      {
          if ((rc = ct_res_info(cmd, CS_NUMDATA, &cols,
CS_UNUSED, NULL))
        != CS_SUCCEED)
        {
          ERRORMSG("[do_ords] ct_res_info failed,
rc="<<rc<<endl);
        }
        else
        {
          DEBUGMSG("[do_ords] cols="<<cols<<endl);
        }

        CT_BND(cmd,1,stx->p_ordP.out.datafmt_c_id,
          stx->ordP.out_ords.s_C_ID);
        CT_BND(cmd,2,stx->p_ordP.out.datafmt_c_last,
          stx->ordP.out_ords.s_C_LAST);
        CT_BND(cmd,3,stx->p_ordP.out.datafmt_c_first,
          stx->ordP.out_ords.s_C_FIRST);
        CT_BND(cmd,4,stx->p_ordP.out.datafmt_c_middle,
          stx->ordP.out_ords.s_C_MIDDLE);
        CT_BND(cmd,5,stx->p_ordP.out.datafmt_c_balance,
          stx->ordP.out_ords.s_C_BALANCE);
        CT_BND(cmd,6,stx->p_ordP.out.datafmt_o_id,
          stx->ordP.out_ords.s_O_ID);
        CT_BND(cmd,7,stx->p_ordP.out.datafmt_o_entry_d,
          s_O_ENTRY_D_base);
        CT_BND(cmd,8,stx-
>p_ordP.out.datafmt_o_carrier_id,
          stx->ordP.out_ords.s_O_CARRIER_ID);
      } else {
        CT_BND(cmd,1,stx-
>p_ordP.out.datafmt_ol_supply_w_id,
          stx->ordP.out_ords.s_OL_SUPPLY_W_ID[ordl]);
        CT_BND(cmd,2,stx->p_ordP.out.datafmt_ol_i_id,
          stx->ordP.out_ords.s_OL_I_ID[ordl]);
        CT_BND(cmd,3,stx->p_ordP.out.datafmt_ol_quantity,
          stx->ordP.out_ords.s_OL_QUANTITY[ordl]);
        CT_BND(cmd,4,stx->p_ordP.out.datafmt_ol_amount,
          stx->ordP.out_ords.s_OL_AMOUNT[ordl]);
        CT_BND(cmd,5,stx-
>p_ordP.out.datafmt_ol_delivery_d,
          stx-
>ordP.out_ords.s_OL_DELIVERY_D_base[ordl]);
      }
      DEBUGMSG("[OS]Fetching"<<endl);
      while ((ret =
ct_fetch(cmd,CS_UNUSED,CS_UNUSED,CS_UNUSED,&ro
ws)) ==
        CS_SUCCEED)
      {
        DEBUGMSG("[do_ords] rows="<<rows<<",
ordl"<<endl);
        if (!second_round) {
```

```
//              stx->ordP.out_ords.s_OL_AMOUNT[ordl] /=
100.0;
            convert_dt(stx->context,
              &stx-
>ordP.out_ords.s_OL_DELIVERY_D_base[ordl],
              (char *)&stx-
>ordP.out_ords.s_OL_DELIVERY_D_time[ordl]);
            ordl++;
            // Rebind before fetching the next order_line
            CT_BND(cmd,1,stx-
>p_ordP.out.datafmt_ol_supply_w_id,
              stx->ordP.out_ords.s_OL_SUPPLY_W_ID[ordl]);
            CT_BND(cmd,2,stx->p_ordP.out.datafmt_ol_i_id,
              stx->ordP.out_ords.s_OL_I_ID[ordl]);
            CT_BND(cmd,3,stx-
>p_ordP.out.datafmt_ol_quantity,
              stx->ordP.out_ords.s_OL_QUANTITY[ordl]);
            CT_BND(cmd,4,stx-
>p_ordP.out.datafmt_ol_amount,
              stx->ordP.out_ords.s_OL_AMOUNT[ordl]);
            CT_BND(cmd,5,stx-
>p_ordP.out.datafmt_ol_delivery_d,
              stx-
>ordP.out_ords.s_OL_DELIVERY_D_base[ordl]);
          }
        }
        if (second_round == 1)
        {
          stx->ordP.out_ords.s_ol_cnt = ordl;
          stx->ordP.out_ords.s_C_BALANCE /= 100.0;
          convert_dt(stx->context,
            &s_O_ENTRY_D_base,
           (char  *)&stx->ordP.out_ords.s_O_ENTRY_D_time);

        }

        second_round++;
        break;
      }
    }

    if (ret != CS_END_DATA)
    {
      (CS_VOID)ct_cancel((CS_CONNECTION *)NULL, cmd,
CS_CANCEL_ALL);
      ERRORMSG("[OrderStatus] ret != CS_END_RESULTS,
ret="<<ret<<", rc="<<rc<<endl);
      terror = 1;
    }

    stx->ordP.out_ords.s_transtatus = terror == 0 ? 0 : -1;
    CopyMemory (&ords->out_ords,&stx-
>ordP.out_ords,sizeof(out_ordstat_struct));
    DEBUGMSG("ord out "<<stx->ordP.in_ords.s_W_ID<<"
"<<
       stx->ordP.in_ords.s_D_ID<<" "<<
       stx->ordP.in_ords.s_C_ID<<endl);
    DEBUGMSG("<<do_ords"<<endl);
    return OK;
}


/*
*****************************************************************
** Name         :  do_stok
** Description :
**             Function calls db to execute stok txn
** Parameters:
**             stok_wrapper*   stock txn structs wrapper
**             void*        stored context
** Returns      :
**             int - return code
** Comments     :
**             Attach to thread's context, call nord sql function
**             then detach from context.
*****************************************************************
*/

extern "C" TPCCSYBASEGLUE_API int
do_stok(stok_wrapper *stok,void *ctx)
{
    syb_con_data_t    *stx = (syb_con_data_t *)ctx;
    CS_INT          rc;
     int i;
     int deadlock = 0;
    int count = 0;
    long result_type = 0;
    int terror = 0,num_cols = 0;
    CS_INT unique[MAX_DISTINCTS];
    static int first_transaction = 1;
    CS_RETCODE     ret=CS_ROW_FAIL;

    DEBUGMSG("starting do_stok"<<endl);
    memcpy (&stx->stoP,stok,sizeof(stok_wrapper));
    DEBUGMSG("stok in,"<<endl<<
       "stok d_id: " <<  stok->in_stok.s_D_ID << endl <<
       "stok w_id: " <<  stok->in_stok.s_W_ID << endl <<
       "stok s_threshold: " <<  stok->in_stok.s_threshold <<
endl);
    stx->stoP.in_stok.w_id_len = CS_SIZEOF(stx-
>stoP.in_stok.s_W_ID);
    stx->stoP.in_stok.d_id_len = CS_SIZEOF(stx-
>stoP.in_stok.s_D_ID);
    stx->stoP.in_stok.threshold_len = CS_SIZEOF(stx-
>stoP.in_stok.s_threshold);
    JUST_RETURN(ct_send(stx->sto_cmd), "ct_send()
failed.");

    DEBUGMSG("back from send"<<endl);

    DEBUGMSG("stok mid,"<<endl<<
       "stok d_id: " <<  stok->in_stok.s_D_ID << endl <<
       "stok w_id: " <<  stok->in_stok.s_W_ID << endl <<
       "stok s_threshold: " <<  stok->in_stok.s_threshold <<
endl);
    /*
    ** Process the results.  Loop while ct_results() returns
CS_SUCCEED.
    */
    while ((rc = ct_results(stx->sto_cmd, &result_type)) ==
CS_SUCCEED)
    {
      switch ((int)result_type)
      {
      case CS_END_RESULTS:
        DEBUGMSG("[SL]CS_END_RESULTS"<<endl);
        CS_INT val;
/*
        if (ex_fetch_data(stx->sto_cmd,&val) != CS_SUCCEED)
        {
          ERRORMSG("[Delivery]: ex_fetch_data()
failed"<<endl);
          return ret;
        }
*/
        if (do_rtn(ctx,stx->sto_cmd,&val) != CS_SUCCEED)
        {
          ERRORMSG("[Delivery]: do_rtn() failed"<<endl);
          return ret;
        }
        ret = CS_END_DATA;
        break;
      case CS_CMD_SUCCEED:
        DEBUGMSG("[SL]CS_CMD_SUCCEED"<<endl);
        /*
        ** This means no rows were returned.
        */
        break;

      case CS_CMD_DONE:
        DEBUGMSG("[SL]CS_CMD_DONE"<<endl);
        /*
        ** This means we're done with one result set.
        */
        break;
```

```
      case CS_STATUS_RESULT:
        DEBUGMSG("CS_STATUS_RESULT"<<endl);
        CS_INT v;
/*
        if (ex_fetch_data(stx->sto_cmd,&v) != CS_SUCCEED)
        {
          ERRORMSG("[StockLevel]: ex_fetch_data()
failed"<<endl);
          return ret;
        }
*/
        if (do_rtn(ctx,stx->sto_cmd,&v) != CS_SUCCEED)
        {
          ERRORMSG("[StockLevel]: do_rtn() failed"<<endl);
          return ret;
        }
        if (v == 1)
        {
          ERRORMSG("[SL]Transaction error, return
code="<<v);
          terror = 1;
        }
        DEBUGMSG("[SL]CS_STATUS_RESULT,V="<<v);
      break;
      case CS_CMD_FAIL:
        DEBUGMSG("CS_CMD_FAIL"<<endl);
        /*
        ** This means that the server encountered an error
while
        ** processing our command.
        */
        DEBUGMSG("[StockLevel] ct_results() returned
CMD_FAIL"<<endl);
        terror = 1;
        break;

      case CS_ROW_RESULT:
        DEBUGMSG("[SL]CS_ROW_RESULT"<<endl);
        //bind to the low count db var

        //bind low_count from database only once
//      if (first_transaction)
//      {
          if (ct_bind(stx->sto_cmd,1,&stx-
>p_stoP.out.datafmt_low_count,(BYTE*)&stok-
>out_stok.s_low_stock,(CS_INT *)NULL,(CS_SMALLINT
*)NULL) != CS_SUCCEED)
          {
            DEBUGMSG("[SL]error from ct_bind()"<< endl);
            terror = 1;
            break;
          }
//        first_transaction = 0;
//      }
          while ((ret = ct_fetch(stx-
>sto_cmd,CS_UNUSED,CS_UNUSED,CS_UNUSED,NULL))
==
            CS_SUCCEED)
          {
            if (count > 0 )
            {
              for (i = 0; i < count ; i++)
              {
                if ( stok->out_stok.s_low_stock == unique[i] )
                  break;
              }
              if ( i == count) {
                unique[count++] = stok->out_stok.s_low_stock;
              }

            }
            else
            {
              unique[count++] = stok->out_stok.s_low_stock;
            }
          }
          DEBUGMSG("[SL]ct_fetch, ret="<<ret<<endl);
      }
    }

    if ( count == MAX_DISTINCTS )
    {
      ERRORMSG("Number of Distincts from the stock_level
stored proc exceeded:"<<MAX_DISTINCTS);
      ERRORMSG("Need to recompile the client with new
maximum");

      DEBUGMSG("Number of Distincts from the stock_level
stored proc exceeded:"<<MAX_DISTINCTS);
      DEBUGMSG("Need to recompile the client with new
maximum");

      terror = 1;
    }


    if (ret != CS_END_DATA)
    {
//    (CS_VOID)ct_cmd_drop(stx->cmd);
      (CS_VOID)ct_cancel((CS_CONNECTION *)NULL, stx-
>sto_cmd, CS_CANCEL_ALL);
      ERRORMSG("[StockLevel] ret != CS_END_RESULTS,
ret="<<ret<<", rc="<<rc<<endl);
      ERRORMSG(endl<<"[StockLevel]"<<endl<<
        "s_W_ID "<<stx->stoP.in_stok.s_W_ID<<endl<<
        "s_D_ID "<<stx->stoP.in_stok.s_D_ID<<endl<<
        "s_threshold "<<stx->stoP.in_stok.s_threshold<<endl);
      terror = 1;
    }

    stx->stoP.out_stok.s_low_stock = count;
    stx->stoP.out_stok.s_transtatus = terror == 0 ? 0 : -1;

    memcpy (stok,&stx->stoP,sizeof(stok_wrapper));
    DEBUGMSG("stok out,"<<endl<<
      "stok d_id: " <<  stok->in_stok.s_D_ID << endl <<
      "stok w_id: " <<  stok->in_stok.s_W_ID << endl <<
      "stok s_threshold: " <<  stok->in_stok.s_threshold <<
endl);
    DEBUGMSG("Finished do_stok s_transtatus: "<<stok-
>out_stok.s_transtatus<<endl);
    return OK;
}


void get_client_msgs(CS_CONNECTION *connection)
{
    CS_INT    rc;
    CS_INT    msgno=1;
    CS_INT    numOfMessages=0;

    CS_CLIENTMSG  clientmsg;


    rc = ct_diag (connection, CS_STATUS,
CS_CLIENTMSG_TYPE, CS_UNUSED, &numOfMessages);
    if  (rc != CS_SUCCEED)
    {
      ERRORMSG("CT_DIAG CS_STATUS
CLIENTMSG_TYPE failed,
numOfMessages:"<<numOfMessages);
      DEBUGMSG("CT_DIAG CS_STATUS
CLIENTMSG_TYPE failed,
numOfMessages:"<<numOfMessages);

    }
    else if (numOfMessages > 0)
    {
      for (int cnt = 1;  cnt <= numOfMessages; ++cnt)
      {
        rc = ct_diag  (connection, CS_GET,
CS_CLIENTMSG_TYPE ,msgno, &clientmsg);
        if ( rc  !=  CS_SUCCEED )
        {
```

```
            ERRORMSG("CT_DIAG CS_GET
CS_CLIENTMSG_TYPE  failed."<<endl);
            DEBUGMSG("CT_DIAG CS_GET
CS_CLIENTMSG_TYPE  failed."<<endl);
            return;
          }

          DEBUGMSG("Client
Message:"<<clientmsg.status<<endl);
          DEBUGMSG("Client
Severity:"<<clientmsg.severity<<endl);

          DEBUGMSG("Message
Number:"<<clientmsg.msgnumber<<endl);
          DEBUGMSG("client
message:"<<clientmsg.msgstring<<endl);

          ERRORMSG("Message
Number:"<<clientmsg.msgnumber<<endl);
          ERRORMSG("client
message:"<<clientmsg.msgstring<<endl);
        }
      }
       else
       {
         DEBUGMSG("No Message!"<<endl);
         ERRORMSG("No Message!"<<endl);
       }

      return;
}
```

tpccSybaseGlue.h

```
// The following ifdef block is the standard way of creating
macros which make exporting
// from a DLL simpler. All files within this DLL are compiled
with the TPCCSYBASEGLUE_EXPORTS
// symbol defined on the command line. this symbol should
not be defined on any project
// that uses this DLL. This way any other project whose
source files include this file see
// TPCCSYBASEGLUE_API functions as being imported from
a DLL, whereas this DLL sees symbols
// defined with this macro as being exported.
#ifndef _TPCCSYBASEGLUE_H_
#define _TPCCSYBASEGLUE_H_
#ifdef TPCCSYBASEGLUE_EXPORTS
#define TPCCSYBASEGLUE_API __declspec(dllexport)
#else
#define TPCCSYBASEGLUE_API __declspec(dllimport)
#endif

#ifndef SPGENERAL
  #define SPGENERAL
#endif

extern int debugFlag;

#include "tpcc.h"

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <cspublic.h>
#include <ctpublic.h>

////////////////////////////////////////////////////////////////
// Sybase Defines
////////////////////////////////////////////////////////////////

extern CS_RETCODE   ret;

CS_RETCODE CS_PUBLIC cserror_cb PROTOTYPE((
  CS_CONTEXT *context,
  CS_CLIENTMSG *errmsg
  ));
CS_RETCODE CS_PUBLIC clientmsg_cb PROTOTYPE((
  CS_CONTEXT *context,
  CS_CONNECTION *connection,
  CS_CLIENTMSG *errmsg
  ));
CS_RETCODE CS_PUBLIC servermsg_cb PROTOTYPE((
  CS_CONTEXT *context,
  CS_CONNECTION *connection,
  CS_SERVERMSG *srvmsg
  ));

#define    NORMAL_EXIT (0)
#define    ERROR_EXIT (-1)
#define    USER         "user_name"
#define    PASSWORD     "server_password"

////////////////////////////////////////////////////////////////
// Error/Debug log file defines
////////////////////////////////////////////////////////////////
extern ofstream debugStream;
extern ofstream errorStream;

extern CRITICAL_SECTION debugMutex;
extern CRITICAL_SECTION errorMutex;

////////////////////////////////////////////////////////////////
// Sybase Marcos Defines
////////////////////////////////////////////////////////////////

#define EXIT_ON_FAIL(context, ret, str) \
  { if (ret != CS_SUCCEED) \
  { \
    ERRORMSG("Fatal error: "<<str<<endl);  \
    if (context != (CS_CONTEXT *) NULL) \
    { \
      (CS_VOID) ct_exit(context, CS_FORCE_EXIT); \
      (CS_VOID) cs_ctx_drop(context); \
    } \
    return -1; \
  } }

#define JUST_RETURN(ret,str) \
  { if (ret != CS_SUCCEED) \
  { \
    ERRORMSG("Fatal error: "<<str<<endl);  \
    return -1; \
  } }

#define RTN_ON_FAIL(context, ret, str) \
  { if (ret != CS_SUCCEED) \
  { \
    ERRORMSG("Fatal error: "<<str<<endl);  \
    if (context != (CS_CONTEXT *) NULL) \
    { \
      (CS_VOID) ct_exit(context, CS_FORCE_EXIT); \
      (CS_VOID) cs_ctx_drop(context); \
    } \
    return; \
  } }

#define CT_CMD(CMD,TYPE,X)                  \
    if ((rc = ct_command(CMD,TYPE,X, \
        CS_NULLTERM,CS_NO_RECOMPILE)) !=
CS_SUCCEED)\
    {                                \
    ERRORMSG("cannot execute "<<#X<<", rc="<<rc<<endl); \
        return;                          \
    }

#define _CT_CMD(CMD,TYPE,X)                 \
    if ((rc = ct_command(CMD,TYPE,X, \
        CS_NULLTERM,CS_NO_RECOMPILE)) !=
CS_SUCCEED)\
    {                                \
```

```
    ERRORMSG("cannot execute "<<#X<<",
rc="<<rc<<endl);  \
    return -1;                  \
  }

#define CT_BND(CMD,COL,X,Y) \
  if (ct_bind(CMD,COL,&X,(BYTE*)&Y,(CS_INT
*)NULL,(CS_SMALLINT *)NULL) != CS_SUCCEED) \
  { \
    ERRORMSG("["<<#X<<"] error from ct_bind()"<< endl); \
    terror = 1; \
    break; \
  }

#define _CT_BND(CMD,COL,X,Y) \
  if (ct_bind(CMD,COL,&X,(BYTE*)&Y,(CS_INT
*)NULL,(CS_SMALLINT *)NULL) != CS_SUCCEED) \
  { \
    ERRORMSG("["<<#X<<"] error from ct_bind()"<< endl); \
    return -1; \
  }

#define DTAFMT(X,TYPE,PARM) \
    memset(&X,0,sizeof(CS_DATAFMT)); \
    X.datatype  = TYPE;          \
    X.maxlength  = CS_UNUSED;    \
    X.status     = CS_INPUTVALUE;  \
    strcpy(X.name,PARM);       \
    X.namelen = CS_NULLTERM;   \
    X.locale = NULL;

#define DTAFMT_TST(X,TYPE,PARM,LEN)\
    memset(&X,0,sizeof(CS_DATAFMT)); \
    X.datatype  = TYPE;          \
    X.maxlength  = LEN;    \
    X.status     = CS_INPUTVALUE;  \
    strcpy(X.name,PARM);       \
    X.namelen = CS_NULLTERM;   \
    X.locale = NULL;

#define DTAFMT_BND(X,TYPE,T,CNT) \
    memset(&X,0,sizeof(CS_DATAFMT)); \
    X.datatype  = TYPE;          \
    X.status     = CS_RETURN;   \
    X.format = CS_FMT_UNUSED ;      \
    X.locale = NULL;            \
    X.maxlength  = CS_SIZEOF(T);

#define DTAFMT_CHAR_BND(X,TYPE,CNT,LEN)   \
    memset(&X,0,sizeof(CS_DATAFMT)); \
    X.datatype  = TYPE;          \
    X.status     = CS_RETURN;   \
    X.format = CS_FMT_NULLTERM|CS_FMT_PADBLANK;
      \
    X.locale = NULL;            \
    X.maxlength  = LEN;

#define DTAFMT_CHAR(X,TYPE,PARM,LEN)\
    memset(&X,0,sizeof(CS_DATAFMT)); \
    X.datatype  = TYPE;          \
    X.maxlength  = LEN;    \
    X.status     = CS_INPUTVALUE;  \
    strcpy(X.name,PARM);          \
    X.namelen = CS_NULLTERM;


// STRLEN() -- strlen() that returns 0 for NULL pointers.
#define STRLEN(str) ( ((str) == NULL) ? 0 : (strlen(str)) )

//#define TIMING  1
extern FILE *respTimes;
struct txn
{
  short txnType;
  struct  _timeb        startTime;
  struct  _timeb        endTime;
  short padding;
};

////////////////////////////////////////////////////////////////
// Registery Values
////////////////////////////////////////////////////////////////
#define    DB_USER_NAME             "dbUserName"
#define    DB_USER_PASSWORD         "dbPassword"
#define    DB_NAME                  "dbName"
#define    ENGINE_APP_NAME          "engineApp"

extern char userName[16];
extern char userPassword[16];
extern char dbName[32];


extern HKEY  registryKey;
extern DWORD regType;
extern char  value[MAX_STRING_LEN];
extern char  msgstr[4096];
extern DWORD regValueSize;


////////////////////////////////////////////////////
// Sybase Specific Errors
////////////////////////////////////////////////////

#define    ERR_CS_CTX_ALLOC    -999
#define    ERR_CT_INIT         -998

////////////////////////////////////////////////////
// Sybase structs
////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////
// Structure defines
////////////////////////////////////////////////////////////////

struct in_items_t
{
  CS_DATAFMT  datafmt_supply_w_id;
  CS_DATAFMT  datafmt_i_id;
  CS_DATAFMT  datafmt_qty;
};

struct out_items_t
{
  CS_DATAFMT  datafmt_i_name;
  CS_DATAFMT  datafmt_i_price;
  CS_DATAFMT  datafmt_s_quantity;
  CS_DATAFMT  datafmt_b_g;

};

struct in_nor_t
{
  CS_DATAFMT  datafmt_w_id;
  CS_DATAFMT  datafmt_d_id;
  CS_DATAFMT  datafmt_c_id;
  CS_DATAFMT  datafmt_ol_cnt;
  struct in_items_t ol[15];
};

struct out_nor_t
{
  struct out_items_t ol;
  CS_DATAFMT  datafmt_w_tax;
  CS_DATAFMT  datafmt_d_tax;
  CS_DATAFMT  datafmt_o_id;
  CS_DATAFMT  datafmt_c_last;
  CS_DATAFMT  datafmt_c_discount;
  CS_DATAFMT  datafmt_c_credit;
  CS_DATAFMT  datafmt_o_entry_d;
};

struct in_ord_by_id_t
{
  CS_DATAFMT  datafmt_w_id;
  CS_DATAFMT  datafmt_d_id;
  CS_DATAFMT  datafmt_c_id;
};
```

```
struct in_ord_by_name_t
{
   CS_DATAFMT  datafmt_w_id;
   CS_DATAFMT  datafmt_d_id;
   CS_DATAFMT  datafmt_c_last;
};

struct out_ord_t
{
   CS_DATAFMT  datafmt_ol_supply_w_id;
   CS_DATAFMT  datafmt_ol_i_id;
   CS_DATAFMT  datafmt_ol_quantity;
   CS_DATAFMT  datafmt_ol_amount;
   CS_DATAFMT  datafmt_ol_delivery_d;
   CS_DATAFMT  datafmt_c_id;
   CS_DATAFMT  datafmt_c_last;
   CS_DATAFMT  datafmt_c_first;
   CS_DATAFMT  datafmt_c_middle;
   CS_DATAFMT  datafmt_c_balance;
   CS_DATAFMT  datafmt_o_id;
   CS_DATAFMT  datafmt_o_entry_d;
   CS_DATAFMT  datafmt_o_carrier_id;
};

struct in_sto_t
{
   CS_DATAFMT  datafmt_w_id;
   CS_DATAFMT  datafmt_d_id;
   CS_DATAFMT  datafmt_threshold;
};

struct out_sto_t
{
   CS_DATAFMT  datafmt_low_count;
};

struct in_pay_by_id_t
{
   CS_DATAFMT  datafmt_w_id;
   CS_DATAFMT  datafmt_c_w_id;
   CS_DATAFMT  datafmt_h_amount_cents;
   CS_DATAFMT  datafmt_d_id;
   CS_DATAFMT  datafmt_c_d_id;
   CS_DATAFMT  datafmt_c_id;
};

struct in_pay_by_name_t
{
   CS_DATAFMT  datafmt_w_id;
   CS_DATAFMT  datafmt_c_w_id;
   CS_DATAFMT  datafmt_h_amount_cents;
   CS_DATAFMT  datafmt_d_id;
   CS_DATAFMT  datafmt_c_d_id;
   CS_DATAFMT  datafmt_c_last;
};

struct out_pay_t
{
   CS_DATAFMT  datafmt_c_id;
   CS_DATAFMT  datafmt_c_last;
   CS_DATAFMT  datafmt_h_date;
   CS_DATAFMT  datafmt_w_street_1;
   CS_DATAFMT  datafmt_w_street_2;
   CS_DATAFMT  datafmt_w_city;
   CS_DATAFMT  datafmt_w_state;
   CS_DATAFMT  datafmt_w_zip;
   CS_DATAFMT  datafmt_d_street_1;
   CS_DATAFMT  datafmt_d_street_2;
   CS_DATAFMT  datafmt_d_city;
   CS_DATAFMT  datafmt_d_state;
   CS_DATAFMT  datafmt_d_zip;
   CS_DATAFMT  datafmt_c_first;
   CS_DATAFMT  datafmt_c_middle;
   CS_DATAFMT  datafmt_c_street_1;
   CS_DATAFMT  datafmt_c_street_2;
   CS_DATAFMT  datafmt_c_city;
   CS_DATAFMT  datafmt_c_state;
   CS_DATAFMT  datafmt_c_zip;
   CS_DATAFMT  datafmt_c_phone;
   CS_DATAFMT  datafmt_c_since;
   CS_DATAFMT  datafmt_c_credit;
   CS_DATAFMT  datafmt_c_credit_lim;
   CS_DATAFMT  datafmt_c_discount;
   CS_DATAFMT  datafmt_c_balance;
   CS_DATAFMT  datafmt_c_data;
};

struct in_dvy_t
{
   CS_DATAFMT  datafmt_w_id;
   CS_DATAFMT  datafmt_d_id;
   CS_DATAFMT  datafmt_carrier_id;
};

struct out_dvy_t
{
   CS_DATAFMT  datafmt_o_id;
};

struct nord_pwrapper
{
   struct in_nor_t in;
   struct out_nor_t out;
};

struct paym_pwrapper
{
   struct in_pay_by_id_t in_by_id;
   struct in_pay_by_name_t in_by_name;
   struct out_pay_t out;
};

struct ords_pwrapper
{
   struct in_ord_by_id_t in_by_id;
   struct in_ord_by_name_t in_by_name;
   struct out_ord_t out;
};

struct stok_pwrapper
{
   struct in_sto_t in;
   struct out_sto_t out;
};

struct dlvy_pwrapper
{
   struct in_dvy_t in;
   struct out_dvy_t out;
};

struct syb_con_data_t
{
   CS_CONTEXT     *context;
   CS_CONNECTION *conn;
   CS_COMMAND      *nor_local_cmd;
   CS_COMMAND      *nor_remote_cmd;
   CS_COMMAND      *pay_id_cmd;
   CS_COMMAND      *pay_name_cmd;
   CS_COMMAND      *ord_id_cmd;
   CS_COMMAND      *ord_name_cmd;
   CS_COMMAND      *sto_cmd;
   CS_COMMAND      *del_cmd;

   CS_DATAFMT     datafmt_rtn;


   nord_wrapper norP;
   ords_wrapper ordP;
   stok_wrapper stoP;
   paym_wrapper payP;
   dlvy_wrapper delP;

   nord_pwrapper p_norP;
   ords_pwrapper p_ordP;
```

```
   stok_pwrapper p_stoP;
   paym_pwrapper p_payP;
   dlvy_pwrapper p_delP;

   int tid;
   CS_RETCODE sticky_binds;
   CS_SMALLINT nonullind;
   CS_SMALLINT nullind;
   char engine_app[32];
};

////////////////////////////////////////////////////////////////////
// Sybase Glue Function Prototypes
////////////////////////////////////////////////////////////////////
extern "C" TPCCSYBASEGLUE_API int connect_db(char
*dbName,void **ctx);
extern "C" TPCCSYBASEGLUE_API int getContext(void
**ctx);
extern "C" TPCCSYBASEGLUE_API int detachContext(void
*ctx);
extern "C" TPCCSYBASEGLUE_API int attachContext(void
*ctx);
extern "C" TPCCSYBASEGLUE_API int disconnect_db(void
*ctx);
extern "C" TPCCSYBASEGLUE_API int DBInit(void *ctx);


extern "C" TPCCSYBASEGLUE_API int
do_nord(nord_wrapper *nord,void *ctx);
extern "C" TPCCSYBASEGLUE_API int
do_pymt(paym_wrapper *pymt,void *ctx);
extern "C" TPCCSYBASEGLUE_API int
do_ords(ords_wrapper *ords,void *ctx);
extern "C" TPCCSYBASEGLUE_API int
do_dlvy(dlvy_wrapper *dlvy,void *ctx);
extern "C" TPCCSYBASEGLUE_API int
do_stok(stok_wrapper *stok,void *ctx);

void get_client_msgs(CS_CONNECTION *connection);
#endif // _TPCCSYBASEGLUE_H_
```

10.3. Client Transaction Monitor Code

compreg.cpp

```
// compreg.cpp : Implementation of CCompReg

#include "stdafx.h"
#include "compreg.h"


// CCompReg
```

compreg.h

```
// compreg.h : Declaration of the CCompReg

#pragma once

#include "resource.h"       // main symbols
#include "tpccCom.h"


// CCompReg
class ATL_NO_VTABLE CCompReg :
   public CComObjectRootEx<CComSingleThreadModel>,
   public CComCoClass<CCompReg, &CLSID_CompReg>,
   public IDispatchImpl<IComponentRegistrar,
&IID_IComponentRegistrar, &LIBID_tpccComLib, /*wMajor
=*/ 1, /*wMinor =*/ 0>
{
public:
   CCompReg()
   {
   }

DECLARE_NO_REGISTRY()

BEGIN_COM_MAP(CCompReg)
   COM_INTERFACE_ENTRY(IComponentRegistrar)
   COM_INTERFACE_ENTRY(IDispatch)
END_COM_MAP()

// IComponentRegistrar
public:
    STDMETHOD(Attach)(BSTR bstrPath)
   {
      return S_OK;
   }
   STDMETHOD(RegisterAll)()
   {
      return _AtlComModule.RegisterServer(TRUE);
   }
   STDMETHOD(UnregisterAll)()
   {
      _AtlComModule.UnregisterServer(TRUE);
      return S_OK;
   }
  STDMETHOD(GetComponents)(SAFEARRAY **ppCLSIDs,
SAFEARRAY **ppDescriptions)
   {
      if( ppCLSIDs == NULL || ppDescriptions == NULL )
         return E_POINTER;
      int nComponents = 0;
      for (_ATL_OBJMAP_ENTRY** ppEntry =
_AtlComModule.m_ppAutoObjMapFirst; ppEntry <
_AtlComModule.m_ppAutoObjMapLast; ppEntry++)
      {
         if (*ppEntry != NULL)
         {
            _ATL_OBJMAP_ENTRY* pEntry = *ppEntry;
            if (pEntry->pclsid != NULL)
            {
               LPCTSTR pszDescription = pEntry-
>pfnGetObjectDescription();
               if (pszDescription)
                  nComponents++;
            }
         }
      }
      SAFEARRAYBOUND rgBound[1];
      rgBound[0].lLbound = 0;
      rgBound[0].cElements = nComponents;
      *ppCLSIDs = SafeArrayCreate(VT_BSTR, 1, rgBound);
      if( *ppCLSIDs == NULL )
         return AtlHresultFromLastError();
      *ppDescriptions = SafeArrayCreate(VT_BSTR, 1,
rgBound);
      if( *ppDescriptions == NULL )
         return AtlHresultFromLastError();
      LONG i = 0;
      for (_ATL_OBJMAP_ENTRY** ppEntry =
_AtlComModule.m_ppAutoObjMapFirst; ppEntry <
_AtlComModule.m_ppAutoObjMapLast; ppEntry++)
      {
         if (*ppEntry != NULL)
         {
            _ATL_OBJMAP_ENTRY* pEntry = *ppEntry;
            if (pEntry->pclsid != NULL)
            {
               LPCTSTR pszDescription = pEntry-
>pfnGetObjectDescription();
               if (pszDescription)
               {
                  LPOLESTR pszCLSID;
                  StringFromCLSID(*pEntry->pclsid, &pszCLSID);
                  BSTR pBSTR = OLE2BSTR(pszCLSID);
                  if( pBSTR == NULL )
                  {
                     CoTaskMemFree(pszCLSID);
                     return E_OUTOFMEMORY;
                  }
```

```
            HRESULT hResult =
SafeArrayPutElement(*ppCLSIDs, &i, pBSTR);
            CoTaskMemFree(pszCLSID);
            if( FAILED(hResult) )
               return hResult;
            pBSTR = T2BSTR_EX(pszDescription);
            if( pBSTR == NULL )
            {
               return E_OUTOFMEMORY;
            }
            hResult = SafeArrayPutElement(*ppDescriptions,
&i, pBSTR);
            if( FAILED(hResult) )
               return hResult;
            i++;
          }
        }
      }
    }

    return S_OK;
  }
  STDMETHOD(RegisterComponent)(BSTR bstrCLSID)
  {
    CLSID clsid;
    CLSIDFromString(bstrCLSID, &clsid);
    _AtlComModule.RegisterServer(TRUE, &clsid);
    return S_OK;
  }
  STDMETHOD(UnregisterComponent)(BSTR bstrCLSID)
  {
    CLSID clsid;
    CLSIDFromString(bstrCLSID, &clsid);
    _AtlComModule.UnregisterServer(FALSE, &clsid);
    return S_OK;
  }
};

OBJECT_ENTRY_AUTO(CLSID_CompReg, CCompReg)
```

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

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

#define PROXY_DELEGATION

#include <rpcproxy.h>

#ifdef __cplusplus
extern "C"   {
#endif

EXTERN_PROXY_FILE( tpccCom )


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


DLLDATA_ROUTINES( aProxyFileList, GET_DLL_CLSID )

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

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

dlldatax.c

```
// wrapper for dlldata.c

#ifdef _MERGE_PROXYSTUB // merge proxy stub DLL

#define REGISTER_PROXY_DLL //DllRegisterServer, etc.

#define _WIN32_WINNT 0x0500   //for Win2000, change it to
0x0400 for NT4 or Win95 with DCOM
#define USE_STUBLESS_PROXY   //defined only with MIDL
switch /Oicf

#pragma comment(lib, "rpcns4.lib")
#pragma comment(lib, "rpcrt4.lib")

#define ENTRY_PREFIX  Prx

#include "dlldata.c"
#include "tpccCom_p.c"

#endif //_MERGE_PROXYSTUB
```

dlldatax.h

```
#pragma once

#ifdef _MERGE_PROXYSTUB

extern "C"
{
BOOL WINAPI PrxDllMain(HINSTANCE hInstance, DWORD
dwReason,
   LPVOID lpReserved);
STDAPI PrxDllCanUnloadNow(void);
STDAPI PrxDllGetClassObject(REFCLSID rclsid, REFIID riid,
LPVOID* ppv);
STDAPI PrxDllRegisterServer(void);
STDAPI PrxDllUnregisterServer(void);
}

#endif
```

Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by tpccCom.rc
//
#define IDS_PROJNAME                100
#define IDR_TPCCCOM                 101
#define IDR_TPCC_COM                102

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        201
#define _APS_NEXT_COMMAND_VALUE         32768
#define _APS_NEXT_CONTROL_VALUE         201
#define _APS_NEXT_SYMED_VALUE           103
#endif
#endif
```

stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard
includes
// tpccCom.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"
```

stdafx.h

```
// stdafx.h : include file for standard system include files,
// or project specific include files that are used frequently,
// but are changed infrequently

#pragma once

#ifndef STRICT
#define STRICT
#endif

// Modify the following defines if you have to target a platform
prior to the ones specified below.
// Refer to MSDN for the latest info on corresponding values
for different platforms.
#ifndef WINVER              // Allow use of features specific to
Windows 95 and Windows NT 4 or later.
#define WINVER 0x0400       // Change this to the appropriate
value to target Windows 98 and Windows 2000 or later.
#endif

#ifndef _WIN32_WINNT    // Allow use of features specific to
Windows NT 4 or later.
#define _WIN32_WINNT 0x0400   // Change this to the
appropriate value to target Windows 2000 or later.
#endif

#ifndef _WIN32_WINDOWS    // Allow use of features specific
to Windows 98 or later.
#define _WIN32_WINDOWS 0x0410 // Change this to the
appropriate value to target Windows Me or later.
#endif

#ifndef _WIN32_IE         // Allow use of features specific to IE
4.0 or later.
#define _WIN32_IE 0x0400  // Change this to the appropriate
value to target IE 5.0 or later.
#endif

#define _ATL_APARTMENT_THREADED
#define _ATL_NO_AUTOMATIC_NAMESPACE

#define _ATL_CSTRING_EXPLICIT_CONSTRUCTORS //
some CString constructors will be explicit

// turns off ATL's hiding of some common and often safely
ignored warning messages
#define _ATL_ALL_WARNINGS


#include <comsvcs.h>

#include "resource.h"
```

```
#include <atlbase.h>
#include <atlcom.h>

using namespace ATL;
t
```

tpcc_com.cpp

```
// tpcc_com.cpp : Implementation of Ctpcc_com

#include "stdafx.h"
#include "tpcc_com.h"

#include "..\tpccIsapi\tpcc.h"

#include ".\tpcc_com.h"
#ifdef DB2
#include <db2tpcc.h>
#endif
#ifdef ORA
#include <ora_tpcc.h>
#endif

#ifdef DEBUG
   int debugFlag = 1;
#else
   int debugFlag = 0;
#endif

// Ctpcc_com
void handle_exit()
{
   ERRORMSG("Exit caught"<<endl);
   DBInit(NULL);
}

HRESULT Ctpcc_com::Activate()
{
   HRESULT hr = GetObjectContext(&m_spObjectContext);
   if (SUCCEEDED(hr))
   {
      DEBUGMSG("Object assigned to thread."<<endl);
      return S_OK;
   }
   return hr;
}

BOOL Ctpcc_com::CanBePooled()
{
   DEBUGMSG("CanBePooled() returning true"<<endl);
   return TRUE;
}
```

```
void Ctpcc_com::Deactivate()
{
   DEBUGMSG("deactivated() releasing object back into
pool"<<endl);
   m_spObjectContext.Release();
}

/*
*****************************************************************
** Name        :  doSetComplete
** Description :
**             Release object back into com pool
** Parameters:
** Returns     :
**             int - return code
** Comments    :
**             Calls SetComplete on the object that the com
**             pool manager returned to the caller(isapi thread)
*****************************************************************
*/
STDMETHODIMP Ctpcc_com::doSetComplete(void)
{
   // TODO: Add your implementation code here
   HRESULT hres = m_spObjectContext->SetComplete();
   if (SUCCEEDED(hres))
   {
      DEBUGMSG("SetComplete successful. object bit set to
release object into pool."<<endl);
   }
   else
   {
      DEBUGMSG("SetComplete failed. object bit set to
release object into pool."<<endl);
      ERRORMSG("SetComplete() failed,
code:"<<HRESULT_CODE(hres)<<"
facility:"<<HRESULT_FACILITY(hres)<<"
hres:"<<hex<<hres<<endl);
   }

   return S_OK;
}

/*
*****************************************************************
** Name        :  doStockLevel
** Description :
**             Call db2 dll entry point to execute txn
** Parameters:
**             int*    size of UCHAR buffer to pay attention to
**             UCHAR**    char buffer that holds txn wrapper
struct
** Returns     :
```

```
**            int - return code
** Comments    :
**
*****************************************************************
*/
STDMETHODIMP Ctpcc_com::doStockLevel(INT *size,
UCHAR **buffer)
{
  stok_wrapper * stok;
  stok = (stok_wrapper *) *buffer;

  if(!connectHandleInUse)
  {
    DEBUGMSG("Setting Context handle in use to
true"<<endl);
    connectHandleInUse = 1;
  }
  else
  {
    DEBUGMSG("Context handle in use."<<endl);
    ERRORMSG("Context handle in use."<<endl);
    return ERR_HANDLE_IN_USE;
  }

  DEBUGMSG("Calling do_stok call using
connectHandle:"<<DEBUGADDRESS(connectHandle)<<"
w_id:"<<stok->in_stok.s_W_ID<<" d_id:"<< stok-
>in_stok.s_D_ID<<
        " s_transtatus:"<<stok->out_stok.s_transtatus<<endl);

  int retry = 0;
  do
  {
    do_stok(stok,connectHandle);
  } while (stok->out_stok.s_transtatus && retry++ < 3);

  DEBUGMSG("Return from do_stok call using
connectHandle:"<<DEBUGADDRESS(connectHandle)<<"
w_id:"<<stok->in_stok.s_W_ID<<" d_id:"<< stok-
>in_stok.s_D_ID<<
        " s_transtatus:"<<stok->out_stok.s_transtatus<<endl);

  DEBUGMSG("Connection handle set to free" <<endl);
  connectHandleInUse = 0;

  return S_OK;
}

/*
*****************************************************************
** Name        :  doNewOrder
** Description :
**            Call db2 dll entry point to execute txn
** Parameters:
**            int*    size of UCHAR buffer to pay attention to
**            UCHAR**    char buffer that holds txn wrapper
struct
** Returns     :
**            int - return code
** Comments    :
**
*****************************************************************
*/
STDMETHODIMP Ctpcc_com::doNewOrder(INT* size,
UCHAR** buffer)
{
  nord_wrapper *nord;
  nord = (nord_wrapper *) *buffer;

  if(!connectHandleInUse)
  {
    DEBUGMSG("Setting Context handle in use to
true"<<endl);
    connectHandleInUse = 1;
  }
  else
  {
    DEBUGMSG("Context handle in use."<<endl);
    ERRORMSG("Context handle in use."<<endl);
    return ERR_HANDLE_IN_USE;
  }

  DEBUGMSG("Calling do_nord call using
connectHandle:"<<DEBUGADDRESS(connectHandle)<<"
w_id:"<<nord->in_nord.s_W_ID<<" d_id:"<< nord-
>in_nord.s_D_ID<<
        " s_transtatus:"<<nord->out_nord.s_transtatus<<endl);

  do_nord(nord,connectHandle);

  DEBUGMSG("Return from do_nord call using
connectHandle:"<<DEBUGADDRESS(connectHandle)<<"
w_id:"<<nord->in_nord.s_W_ID<<" d_id:"<< nord-
>in_nord.s_D_ID<<
        " s_transtatus:"<<nord->out_nord.s_transtatus<<endl);

  DEBUGMSG("Connection handle set to free" <<endl);
  connectHandleInUse = 0;

  return S_OK;
}

/*
*****************************************************************
** Name        :  doPayment
** Description :
**            Call db2 dll entry point to execute txn
** Parameters:
**            int*    size of UCHAR buffer to pay attention to
**            UCHAR**    char buffer that holds txn wrapper
struct
** Returns     :
**            int - return code
** Comments    :
**
*****************************************************************
*/
STDMETHODIMP Ctpcc_com::doPayment(INT* size,
UCHAR** buffer)
{
  paym_wrapper  *pymt;
  pymt = (paym_wrapper *) *buffer;

  if(!connectHandleInUse)
  {
    DEBUGMSG("Setting Context handle in use to
true"<<endl);
    connectHandleInUse = 1;
  }
  else
  {
    DEBUGMSG("Context handle in use."<<endl);
    ERRORMSG("Context handle in use."<<endl);
    return ERR_HANDLE_IN_USE;
  }

  DEBUGMSG("Calling do_pymt call using
connectHandle:"<<DEBUGADDRESS(connectHandle)<<"
w_id:"<<pymt->in_paym.s_W_ID<<" d_id:"<< pymt-
>in_paym.s_D_ID<<
        " s_transtatus:"<<pymt-
>out_paym.s_transtatus<<endl);

  do_pymt(pymt,connectHandle);

  DEBUGMSG("Return from do_pymt call using
connectHandle:"<<DEBUGADDRESS(connectHandle)<<"
w_id:"<<pymt->in_paym.s_W_ID<<" d_id:"<< pymt-
>in_paym.s_D_ID<<
        " s_transtatus:"<<pymt-
>out_paym.s_transtatus<<endl);

  DEBUGMSG("Connection handle set to free" <<endl);
  connectHandleInUse = 0;
```

```
   return S_OK;
}

/*
*********************************************************************
** Name         :  doOrderStatus
** Description :
**            Call db2 dll entry point to execute txn
** Parameters:
**            int*    size of UCHAR buffer to pay attention to
**            UCHAR**    char buffer that holds txn wrapper
struct
** Returns      :
**            int - return code
** Comments     :
**
*********************************************************************
*/

STDMETHODIMP Ctpcc_com::doOrderStatus(INT* size,
UCHAR** buffer)
{
   ords_wrapper *ords;
   ords = (ords_wrapper *) *buffer;

   if(!connectHandleInUse)
   {
      DEBUGMSG("Setting Context handle in use to
true"<<endl);
      connectHandleInUse = 1;
   }
   else
   {
      DEBUGMSG("Context handle in use."<<endl);
      ERRORMSG("Context handle in use."<<endl);
      return ERR_HANDLE_IN_USE;
   }


   DEBUGMSG("Calling do_ords call using
connectHandle:"<<DEBUGADDRESS(connectHandle)<<"
w_id:"<<ords->in_ords.s_W_ID<<" d_id:"<< ords-
>in_ords.s_D_ID<<
         " s_transtatus:"<<ords->out_ords.s_transtatus<<endl);

   int retry = 0;
   do
   {
      do_ords(ords,connectHandle);
   } while (ords->out_ords.s_transtatus && retry++ < 3);

   DEBUGMSG("Return from do_ords call using
connectHandle:"<<DEBUGADDRESS(connectHandle)<<"
w_id:"<<ords->in_ords.s_W_ID<<" d_id:"<< ords-
>in_ords.s_D_ID<<
         " s_transtatus:"<<ords->out_ords.s_transtatus<<endl);

   DEBUGMSG("Connection handle set to free" <<endl);
   connectHandleInUse = 0;

   return S_OK;
}

/*
*********************************************************************
** Name         :  doDBInfo
** Description :
**            Function to test com interface
** Parameters:
** Returns      :
**            int - return code
** Comments     :
**
*********************************************************************
*/

STDMETHODIMP Ctpcc_com::doDBInfo(void)
{
   return S_OK;
}

/*
*********************************************************************
** Name         :  loadLibrary
** Description :
**            Function loads apprioate db library based on
**            registry setting
** Parameters:
** Returns      :
**            int - return code
** Comments     :
**
*********************************************************************
*/

Ctpcc_com::loadLibrary()
{
   DEBUGMSG("Entered loadLibrary function"<<endl);
   //check to see if dbInstance is already loaded
   if(!dbInstance)
   {
      DEBUGMSG("Database dll not loaded. Loading
dll."<<endl);
      if (nullDB)
      {
         DEBUGMSG("Loading "<<dbType << " nulldb dll." <<
endl);
         dbInstance =
LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\nullDB.dll");
         if(dbInstance == NULL)
         {
            DEBUGMSG("Unable to load null db dll,
rc:"<<GetLastError());
            ERRORMSG("Unable to load null db dll,
rc:"<<GetLastError());
            return ERR_NULL_DLL_NOT_LOADED;
         }
         DEBUGMSG(dbType << " nulldb dll loaded"<<endl);
      }
      else if(strcmp(dbType,"DB2") == 0)
      {
         DEBUGMSG("Loading "<<dbType << " dll." << endl);

         dbInstance =
LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccDB2glue.dll");
         if(dbInstance == NULL)
         {
            DEBUGMSG("Unable to load library."<<endl);

            ERRORMSG("Unable to load com dll, rc:" <<
GetLastError() << endl);

            return ERR_DB2_DLL_NOT_LOADED;
         }
         DEBUGMSG(dbType<< " dll loaded"<<endl);
      }
      else if( strcmp(dbType,"ORACLE") == 0 )
      {
         DEBUGMSG("Loading "<<dbType << " dll." << endl);

         dbInstance =
LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccOracleglue.dll");
         if(dbInstance == NULL)
         {
            DEBUGMSG("Unable to load oracle dll"<<endl);
            ERRORMSG("Unable to load oracle dll,
rc:"<<GetLastError()<<endl);
            return ERR_ORACLE_DLL_NOT_LOADED;
         }
         DEBUGMSG(dbType<< " dll loaded"<<endl);
      }
      else if( strcmp(dbType,"SYBASE") == 0 )
      {
         DEBUGMSG("Loading "<<dbType << " dll." << endl);
```

```
        dbInstance =
LoadLibrary("c:\\inetpub\\wwwroot\\tpcc\\tpccSybaseGlue.dll");
        if(dbInstance == NULL)
        {
          DEBUGMSG("Unable to load sybase dll"<<endl);
          ERRORMSG("Unable to load sybase dll,
rc:"<<GetLastError()<<endl);
          return ERR_ORACLE_DLL_NOT_LOADED;
        }
        DEBUGMSG(dbType<< " dll loaded"<<endl);
      }
      else
      {
        DEBUGMSG("Unknown database type
dll:"<<dbType<<endl);
        ERRORMSG("Unknown database type
dll:"<<dbType<<endl);
        return ERR_UNKNOWN_DB;
      }

      //retrieve function addresses from instance loaded.
      DEBUGMSG("Getting do_connection function address
from "<<dbType<<" dll"<<endl);
      if( (do_connection =
(CONNECT_PTR)GetProcAddress(dbInstance,"connect_db"))
== NULL )
        return ERR_CONNECT_ADDRESS_NOT_FOUND;
      DEBUGMSG("do_connection
address:"<<DEBUGADDRESS(do_connection)<<endl);

      DEBUGMSG("Getting do_disconnect function address
from "<<dbType<<" dll"<<endl);
      if( (do_disconnect =
(DISCONNECT_PTR)GetProcAddress(dbInstance,"disconne
ct_db")) == NULL )
        return  ERR_DISCONNECT_ADDRESS_NOT_FOUND;
      DEBUGMSG("do_disconnect
address:"<<DEBUGADDRESS(do_disconnect)<<endl);

      DEBUGMSG("Getting DBInit function address from
"<<dbType<<" dll"<<endl);
      if( (DBInit =
(DBINIT_PTR)GetProcAddress(dbInstance,"DBInit")) ==
NULL )
        return ERR_DBINIT_ADDRESS_NOT_FOUND;
      DEBUGMSG("DBInit
address:"<<DEBUGADDRESS(DBInit)<<endl);

      DEBUGMSG("Getting do_nord function address from
"<<dbType<<" dll"<<endl);
      if( (do_nord = (NORD_PTR)
GetProcAddress(dbInstance,"do_nord")) == NULL)
        return ERR_NORD_ADDRESS_NOT_FOUND;
      DEBUGMSG("do_nord function
address:"<<DEBUGADDRESS(do_nord)<<endl);

      DEBUGMSG("Getting do_pymt function address from
"<<dbType<<" dll"<<endl);
      if( (do_pymt = (PYMT_PTR)
GetProcAddress(dbInstance,"do_pymt")) == NULL)
        return ERR_PYMT_ADDRESS_NOT_FOUND;
      DEBUGMSG("do_pymt function
adddress:"<<DEBUGADDRESS(do_pymt)<<endl);

      DEBUGMSG("Getting do_ords function address from
"<<dbType<<" dll"<<endl);
      if( (do_ords = (ORDS_PTR)
GetProcAddress(dbInstance,"do_ords")) == NULL)
        return ERR_ORDS_ADDRESS_NOT_FOUND;
      DEBUGMSG("do_ords function
address:"<<DEBUGADDRESS(do_ords)<<endl);

      DEBUGMSG("Getting do_stok function address from
"<<dbType<<" dll"<<endl);
      if( (do_stok = (STOK_PTR)
GetProcAddress(dbInstance,"do_stok")) == NULL)
        return ERR_STOK_ADDRESS_NOT_FOUND;
      DEBUGMSG("do_stok function
address:"<<DEBUGADDRESS(do_stok)<<endl);

      DEBUGMSG("All function addresses retrieved
successfully."<<endl);

    }
    return OK;
}

/*
*********************************************************************
** Name         :  readRegistry()
** Description :
**             Function reads registry value
** Parameters:
** Returns     :
**             int - return code
** Comments    :
**             Values retrieved from registery
**             dbName, dbUserName, and dbUserPassword
*********************************************************************
*/

Ctpcc_com::readRegistery()
{
    //open registery key
    HKEY   registeryKey;
    DWORD  regType;
    char   value[MAX_STRING_LEN];
    DWORD  regValue;
    DWORD  regValueSize = MAX_STRING_LEN;

    DEBUGMSG("Entered readRegistery(), opening key:"<<
REGISTERY_SUB_KEY <<endl);
    //open up registery key
    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTERY
_SUB_KEY,0,KEY_READ,&registeryKey) ==
ERROR_SUCCESS)
    {
      DEBUGMSG(REGISTERY_SUB_KEY<<" open, getting
database type from key"<<endl);
      regValueSize = sizeof(value);
      if
(RegQueryValueEx(registeryKey,DB_TYPE,0,&regType,(BY
TE *) &value,&regValueSize)== ERROR_SUCCESS )
        strcpy(dbType,value);
      DEBUGMSG("Database type:"<<dbType<<" from
registery key."<<endl);

      DEBUGMSG("Getting database name from registery
key."<<endl);
      regValueSize = sizeof(value);
      if
(RegQueryValueEx(registeryKey,DB_NAME,0,&regType,(BY
TE *) &value,&regValueSize)== ERROR_SUCCESS )
        strcpy(dbName,value);
      DEBUGMSG("Database name:"<<dbName<<endl);

      DEBUGMSG("Getting null database flag from
key."<<endl);
      regValueSize = sizeof(regValue);

    if(RegQueryValueEx(registeryKey,NULL_DB,0,&regType,(
BYTE *)&regValue,&regValueSize) == ERROR_SUCCESS)
        nullDB = regValue;
      DEBUGMSG("Null database flag:"<<nullDB<<endl);

      return OK;
    }

    DEBUGMSG("Error, unable to open registery key."<<endl);
    return ERR_UNABLE_TO_OPEN_REG;
}

/*
*********************************************************************
** Name         :  connectDB
** Description :
```

```
**            Function connects to the db
** Parameters:
** Returns      :
**            int - return code
** Comments    :
**
*********************************************************************
*/
Ctpcc_com::connectDB()
{
  bool bRetry = false;
  int  count=0;
  DEBUGMSG("Entered connectDB(), checking if object is
connected."<<endl);
  if(!connected)
  {
    DEBUGMSG("Object not connected, calling
do_connection with dbName:"<<dbName<<"
connectHandle:"<<
          DEBUGADDRESS(connectHandle)<<endl);
    if(!connectHandleInUse)
    {
      ERRORMSG("Setting Context handle in use to
true"<<endl);
      connectHandleInUse = 1;

  ERRORMSG("connectHandle:"<<connectHandle<<endl);
      do {
        bRetry = false;
        connected =
do_connection(dbName,&connectHandle);
        ERRORMSG("New
connectHandle:"<<connectHandle<<endl);
        if(connected != OK)
        {
          DEBUGMSG("Object do_connect failed,
rc:"<<connected<<endl);
          ERRORMSG("Object do_connect failed,
rc:"<<connected<<endl);
          bRetry = true;
          if (count++ > 100)
            return connected;
          Sleep(50);
        } else
          bRetry = false;
      } while (bRetry);
      DEBUGMSG("Object connection complete,
connectHandle:"<<DEBUGADDRESS(connectHandle)<<endl
);
      connectHandleInUse = 0;
      return OK;
    }
    else
    {
      DEBUGMSG("Object's connectHandle already in use,
connect failed"<<endl);
      ERRORMSG("Object's connectHandle already in use,
connect failed"<<endl);
      return ERR_HANDLE_IN_USE;
    }
  }
  DEBUGMSG("Object already has connection
established."<<endl);
  return OK;
}
```

tpccCom.cpp

```
// tpccCom.cpp : Implementation of DLL Exports.
//
// Note: COM+ 1.0 Information:
//      Please remember to run Microsoft Transaction Explorer
to install the component(s).
//      Registration is not done by default.

#include "stdafx.h"
#include "resource.h"
#include "tpccCom.h"
#include "compreg.h"
#include "dlldatax.h"

class CtpccComModule : public CAtlDllModuleT<
CtpccComModule >
{
public :
  DECLARE_LIBID(LIBID_tpccComLib)
  DECLARE_REGISTRY_APPID_RESOURCEID(IDR_TPC
CCOM, "{11ED2355-1A27-42F1-ADFF-F201F5E82BCE}")
};

CtpccComModule _AtlModule;

void handle_exit();

// DLL Entry Point
extern "C" BOOL WINAPI DllMain(HINSTANCE hInstance,
DWORD dwReason, LPVOID lpReserved)
{
#ifdef _MERGE_PROXYSTUB
   if (!PrxDllMain(hInstance, dwReason, lpReserved))
      return FALSE;
#endif
  hInstance;
  atexit(handle_exit);
  return _AtlModule.DllMain(dwReason, lpReserved);
}


// Used to determine whether the DLL can be unloaded by
OLE
STDAPI DllCanUnloadNow(void)
{
#ifdef _MERGE_PROXYSTUB
   HRESULT hr = PrxDllCanUnloadNow();
   if (FAILED(hr))
      return hr;
#endif
   return _AtlModule.DllCanUnloadNow();
}


// Returns a class factory to create an object of the requested
type
STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid,
LPVOID* ppv)
{
#ifdef _MERGE_PROXYSTUB
   if (PrxDllGetClassObject(rclsid, riid, ppv) == S_OK)
      return S_OK;
#endif
   return _AtlModule.DllGetClassObject(rclsid, riid, ppv);
}


// DllRegisterServer - Adds entries to the system registry
STDAPI DllRegisterServer(void)
{
   // registers object, typelib and all interfaces in typelib
   HRESULT hr = _AtlModule.DllRegisterServer();
#ifdef _MERGE_PROXYSTUB
   if (FAILED(hr))
      return hr;
   hr = PrxDllRegisterServer();
#endif
  return hr;
}


// DllUnregisterServer - Removes entries from the system
registry
STDAPI DllUnregisterServer(void)
{
  HRESULT hr = _AtlModule.DllUnregisterServer();
#ifdef _MERGE_PROXYSTUB
   if (FAILED(hr))
```

```
    return hr;
  hr = PrxDllRegisterServer();
  if (FAILED(hr))
    return hr;
  hr = PrxDllUnregisterServer();
#endif
  return hr;
}
```

tpccCom.def

```
; tpccCom.def : Declares the module parameters.

LIBRARY      "tpccCom.DLL"

EXPORTS
  DllCanUnloadNow      PRIVATE
  DllGetClassObject   PRIVATE
  DllRegisterServer PRIVATE
  DllUnregisterServer PRIVATE
```

tpcc_com.h

```
// tpcc_com.h : Declaration of the Ctpcc_com

#pragma once
#include "tpccCom.h"
#include "resource.h"// main symbols
#include <comsvcs.h>

#include "..\tpccIsapi\tpcc.h"

#ifdef ORACLE
  #define ORA_NT
  #include <stdio.h>
  #include <time.h>
  #include <ora_tpcc.h>
  #include <plora.h>
#endif

#define NULL_DB        "nullDB"

static HINSTANCE dbInstance        = NULL;

static CRITICAL_SECTION debugMutex;
static CRITICAL_SECTION errorMutex;
static ofstream debugStream;
static ofstream errorStream;


static int comServerID        =   0;
static int debugFileOpen      = 0;
static int errorFileOpen        = 0;
static int nullDB             = 0;
static char dbType[32];
static char dbName[32];

typedef INT (*NORD_PTR)(nord_wrapper *nord,void
*connectHandle);
typedef INT (*PYMT_PTR)(paym_wrapper *pymt,void
*connectHandle);
typedef INT (*ORDS_PTR)(ords_wrapper *ords,void
*connectHandle);
typedef INT (*STOK_PTR)(stok_wrapper *stok,void
*connectHandle);
typedef INT (*CONNECT_PTR)(char *dbName,void
**connectHandle);
typedef INT (*DISCONNECT_PTR)(void *connectHandle);
typedef INT (*DBINIT_PTR)(void *connectHandle);
DBINIT_PTR DBInit;

NORD_PTR   do_nord;
PYMT_PTR do_pymt;
ORDS_PTR   do_ords;
STOK_PTR do_stok;
CONNECT_PTR  do_connection;
DISCONNECT_PTR do_disconnect;

// Ctpcc_com
class ATL_NO_VTABLE Ctpcc_com :
  public CComObjectRootEx<CComMultiThreadModel>,
  public IObjectControl,
  public CComCoClass<Ctpcc_com, &CLSID_tpcc_com>,
  public Itpcc_com
{
public:
  Ctpcc_com()
  {
    int rc          = ERR;
    connected       = 0;
    connectHandleInUse  = 0;

    if(debugFlag)
    {
      if(!debugFileOpen)
      {
        InitializeCriticalSection(&debugMutex);
        //open comLog
        char comLogFile[128];

  sprintf(comLogFile,"C:\\Inetpub\\wwwroot\\tpcc\\comLog_de
bug.txt");
        debugStream.rdbuf( )-
>open(comLogFile,ios_base::in | ios_base::out |
ios_base::app);

        debugFileOpen = 1;

        DEBUGMSG("Debug file open for writing"<<endl);
      }
    }

    //open error log file
    if(!errorFileOpen)
    {
      InitializeCriticalSection(&errorMutex);

      char errorLogFile[128];

  sprintf(errorLogFile,"C:\\Inetpub\\wwwroot\\tpcc\\comLog_er
r.txt");
      errorStream.rdbuf( )->open(errorLogFile,ios_base::in |
ios_base::out | ios_base::trunc);

      errorFileOpen=1;
    }

    //get registry values
    if((rc = readRegistery()) != OK)
    {
      ERRORMSG("Unable to open registery key " <<
REGISTERY_SUB_KEY << " rc:" << rc <<endl);
      DEBUGMSG("Unable to open registery key " <<
REGISTERY_SUB_KEY << " rc:" << rc <<endl);
      return;
    }

    DEBUGMSG("nullDB:" <<nullDB<<"
dbType:"<<dbType<<" dbName:"<<dbName<<endl);

    //load library based on registry
    if( (rc = loadLibrary()) != OK)
    {
      ERRORMSG("load library failure rc:" << rc << endl);
      return;
    }

    DEBUGMSG("dbtype:"<<dbType<<" instance:" <<
DEBUGADDRESS(dbInstance) << " loaded." << endl);

    //connect to db
    EnterCriticalSection(&errorMutex);
    if((rc = connectDB()) != OK)
    {
```

```
        ERRORMSG("unable to connect to db "<<dbName<<"
rc :"<<rc <<endl);
        LeaveCriticalSection(&errorMutex);
        return;
      }
      LeaveCriticalSection(&errorMutex);

      DEBUGMSG("connected to db " <<dbName<< " rc:"<< rc
<< " context:" <<DEBUGADDRESS(connectHandle) << endl);
   }

   DECLARE_PROTECT_FINAL_CONSTRUCT()

   HRESULT FinalConstruct()
   {
      return S_OK;
   }

   void FinalRelease()
   {
   }

DECLARE_REGISTRY_RESOURCEID(IDR_TPCC_COM)

BEGIN_COM_MAP(Ctpcc_com)
   COM_INTERFACE_ENTRY(Itpcc_com)
   COM_INTERFACE_ENTRY(IObjectControl)
END_COM_MAP()

// IObjectControl
public:
   STDMETHOD(Activate)();
   STDMETHOD_(BOOL, CanBePooled)();
   STDMETHOD_(void, Deactivate)();
   CComPtr<IObjectContext> m_spObjectContext;

// Itpcc_com
public:
   STDMETHOD(doStockLevel)(INT *size, UCHAR **buffer);
   STDMETHOD(doNewOrder)(INT* size, UCHAR** buffer);
   STDMETHOD(doPayment)(INT* size, UCHAR** buffer);
   STDMETHOD(doOrderStatus)(INT* size, UCHAR** buffer);
   STDMETHOD(doDBInfo)(void);
   STDMETHOD(doSetComplete)(void);

   int connected;
   int connectHandleInUse;

private:

   void *connectHandle;
   int loadLibrary();
   int readRegistery();
   int connectDB();
   void handle_exit();

};

OBJECT_ENTRY_AUTO(__uuidof(tpcc_com), Ctpcc_com)
```

tpccCom.h

```
/* this ALWAYS GENERATED file contains the definitions for
the interfaces */


 /* File created by MIDL compiler version 6.00.0361 */
/* at Sat Feb 09 14:06:08 2008
 */
/* Compiler settings for .\tpccCom.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 COM_NO_WINDOWS_H
#include "windows.h"
#include "ole2.h"
#endif /*COM_NO_WINDOWS_H*/

#ifndef __tpccCom_h__
#define __tpccCom_h__

#if defined(_MSC_VER) && (_MSC_VER >= 1020)
#pragma once
#endif

/* Forward Declarations */

#ifndef __IComponentRegistrar_FWD_DEFINED__
#define __IComponentRegistrar_FWD_DEFINED__
typedef interface IComponentRegistrar IComponentRegistrar;
#endif   /* __IComponentRegistrar_FWD_DEFINED__ */


#ifndef __Itpcc_com_FWD_DEFINED__
#define __Itpcc_com_FWD_DEFINED__
typedef interface Itpcc_com Itpcc_com;
#endif   /* __Itpcc_com_FWD_DEFINED__ */


#ifndef __CompReg_FWD_DEFINED__
#define __CompReg_FWD_DEFINED__

#ifdef __cplusplus
typedef class CompReg CompReg;
#else
typedef struct CompReg CompReg;
#endif /* __cplusplus */

#endif   /* __CompReg_FWD_DEFINED__ */


#ifndef __tpcc_com_FWD_DEFINED__
#define __tpcc_com_FWD_DEFINED__

#ifdef __cplusplus
typedef class tpcc_com tpcc_com;
#else
typedef struct tpcc_com tpcc_com;
#endif /* __cplusplus */

#endif   /* __tpcc_com_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 * );

#ifndef __IComponentRegistrar_INTERFACE_DEFINED__
#define __IComponentRegistrar_INTERFACE_DEFINED__

/* interface IComponentRegistrar */
/* [unique][helpstring][dual][uuid][object] */


EXTERN_C const IID IID_IComponentRegistrar;

#if defined(__cplusplus) && !defined(CINTERFACE)

    MIDL_INTERFACE("a817e7a2-43fa-11d0-9e44-
00aa00b6770a")
    IComponentRegistrar : public IDispatch
    {
    public:
        virtual /* [id] */ HRESULT STDMETHODCALLTYPE
Attach(
            /* [in] */ BSTR bstrPath) = 0;

        virtual /* [id] */ HRESULT STDMETHODCALLTYPE
RegisterAll( void) = 0;

        virtual /* [id] */ HRESULT STDMETHODCALLTYPE
UnregisterAll( void) = 0;

        virtual /* [id] */ HRESULT STDMETHODCALLTYPE
GetComponents(
            /* [out] */ SAFEARRAY * *pbstrCLSIDs,
            /* [out] */ SAFEARRAY * *pbstrDescriptions) = 0;

        virtual /* [id] */ HRESULT STDMETHODCALLTYPE
RegisterComponent(
            /* [in] */ BSTR bstrCLSID) = 0;

        virtual /* [id] */ HRESULT STDMETHODCALLTYPE
UnregisterComponent(
            /* [in] */ BSTR bstrCLSID) = 0;

    };

#else /* C style interface */

    typedef struct IComponentRegistrarVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE
*QueryInterface )(
            IComponentRegistrar * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void **ppvObject);

        ULONG ( STDMETHODCALLTYPE *AddRef )(
            IComponentRegistrar * This);

        ULONG ( STDMETHODCALLTYPE *Release )(
            IComponentRegistrar * This);

        HRESULT ( STDMETHODCALLTYPE
*GetTypeInfoCount )(
            IComponentRegistrar * This,
            /* [out] */ UINT *pctinfo);

        HRESULT ( STDMETHODCALLTYPE *GetTypeInfo )(
            IComponentRegistrar * This,
            /* [in] */ UINT iTInfo,
            /* [in] */ LCID lcid,
            /* [out] */ ITypeInfo **ppTInfo);

        HRESULT ( STDMETHODCALLTYPE
*GetIDsOfNames )(
            IComponentRegistrar * This,
            /* [in] */ REFIID riid,
            /* [size_is][in] */ LPOLESTR *rgszNames,
            /* [in] */ UINT cNames,
            /* [in] */ LCID lcid,
            /* [size_is][out] */ DISPID *rgDispId);

        /* [local] */ HRESULT ( STDMETHODCALLTYPE
*Invoke )(
            IComponentRegistrar * This,
            /* [in] */ DISPID dispIdMember,
            /* [in] */ REFIID riid,
            /* [in] */ LCID lcid,
            /* [in] */ WORD wFlags,
            /* [out][in] */ DISPPARAMS *pDispParams,
            /* [out] */ VARIANT *pVarResult,
            /* [out] */ EXCEPINFO *pExcepInfo,
            /* [out] */ UINT *puArgErr);

        /* [id] */ HRESULT ( STDMETHODCALLTYPE *Attach )(
            IComponentRegistrar * This,
            /* [in] */ BSTR bstrPath);

        /* [id] */ HRESULT ( STDMETHODCALLTYPE
*RegisterAll )(
            IComponentRegistrar * This);

        /* [id] */ HRESULT ( STDMETHODCALLTYPE
*UnregisterAll )(
            IComponentRegistrar * This);

        /* [id] */ HRESULT ( STDMETHODCALLTYPE
*GetComponents )(
            IComponentRegistrar * This,
            /* [out] */ SAFEARRAY * *pbstrCLSIDs,
            /* [out] */ SAFEARRAY * *pbstrDescriptions);

        /* [id] */ HRESULT ( STDMETHODCALLTYPE
*RegisterComponent )(
            IComponentRegistrar * This,
            /* [in] */ BSTR bstrCLSID);

        /* [id] */ HRESULT ( STDMETHODCALLTYPE
*UnregisterComponent )(
            IComponentRegistrar * This,
            /* [in] */ BSTR bstrCLSID);

        END_INTERFACE
    } IComponentRegistrarVtbl;

    interface IComponentRegistrar
    {
        CONST_VTBL struct IComponentRegistrarVtbl *lpVtbl;
    };



#ifdef COBJMACROS


#define
IComponentRegistrar_QueryInterface(This,riid,ppvObject)  \
    (This)->lpVtbl -> QueryInterface(This,riid,ppvObject)

#define IComponentRegistrar_AddRef(This) \
    (This)->lpVtbl -> AddRef(This)

#define IComponentRegistrar_Release(This)   \
    (This)->lpVtbl -> Release(This)


#define IComponentRegistrar_GetTypeInfoCount(This,pctinfo)
  \
    (This)->lpVtbl -> GetTypeInfoCount(This,pctinfo)

#define
IComponentRegistrar_GetTypeInfo(This,iTInfo,lcid,ppTInfo)  \
    (This)->lpVtbl -> GetTypeInfo(This,iTInfo,lcid,ppTInfo)
```

```
#define
IComponentRegistrar_GetIDsOfNames(This,riid,rgszNames,
cNames,lcid,rgDispId)\
    (This)->lpVtbl ->
GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)

#define
IComponentRegistrar_Invoke(This,dispIdMember,riid,lcid,wFl
ags,pDispParams,pVarResult,pExcepInfo,puArgErr) \
    (This)->lpVtbl ->
Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVar
Result,pExcepInfo,puArgErr)


#define IComponentRegistrar_Attach(This,bstrPath) \
    (This)->lpVtbl -> Attach(This,bstrPath)

#define IComponentRegistrar_RegisterAll(This)  \
    (This)->lpVtbl -> RegisterAll(This)

#define IComponentRegistrar_UnregisterAll(This)  \
    (This)->lpVtbl -> UnregisterAll(This)

#define
IComponentRegistrar_GetComponents(This,pbstrCLSIDs,pb
strDescriptions) \
    (This)->lpVtbl ->
GetComponents(This,pbstrCLSIDs,pbstrDescriptions)

#define
IComponentRegistrar_RegisterComponent(This,bstrCLSID)  \
    (This)->lpVtbl -> RegisterComponent(This,bstrCLSID)

#define
IComponentRegistrar_UnregisterComponent(This,bstrCLSID)
   \
    (This)->lpVtbl -> UnregisterComponent(This,bstrCLSID)

#endif /* COBJMACROS */


#endif   /* C style interface */



/* [id] */ HRESULT STDMETHODCALLTYPE
IComponentRegistrar_Attach_Proxy(
    IComponentRegistrar * This,
    /* [in] */ BSTR bstrPath);


void __RPC_STUB IComponentRegistrar_Attach_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [id] */ HRESULT STDMETHODCALLTYPE
IComponentRegistrar_RegisterAll_Proxy(
    IComponentRegistrar * This);


void __RPC_STUB IComponentRegistrar_RegisterAll_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [id] */ HRESULT STDMETHODCALLTYPE
IComponentRegistrar_UnregisterAll_Proxy(
    IComponentRegistrar * This);


void __RPC_STUB
IComponentRegistrar_UnregisterAll_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [id] */ HRESULT STDMETHODCALLTYPE
IComponentRegistrar_GetComponents_Proxy(
    IComponentRegistrar * This,
    /* [out] */ SAFEARRAY * *pbstrCLSIDs,
    /* [out] */ SAFEARRAY * *pbstrDescriptions);


void __RPC_STUB
IComponentRegistrar_GetComponents_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [id] */ HRESULT STDMETHODCALLTYPE
IComponentRegistrar_RegisterComponent_Proxy(
    IComponentRegistrar * This,
    /* [in] */ BSTR bstrCLSID);


void __RPC_STUB
IComponentRegistrar_RegisterComponent_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [id] */ HRESULT STDMETHODCALLTYPE
IComponentRegistrar_UnregisterComponent_Proxy(
    IComponentRegistrar * This,
    /* [in] */ BSTR bstrCLSID);


void __RPC_STUB
IComponentRegistrar_UnregisterComponent_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);



#endif   /*
__IComponentRegistrar_INTERFACE_DEFINED__ */


#ifndef __Itpcc_com_INTERFACE_DEFINED__
#define __Itpcc_com_INTERFACE_DEFINED__

/* interface Itpcc_com */
/* [unique][helpstring][uuid][object] */


EXTERN_C const IID IID_Itpcc_com;

#if defined(__cplusplus) && !defined(CINTERFACE)

    MIDL_INTERFACE("5B4FA473-2E68-4D79-A626-
F38B30B8196E")
    Itpcc_com : public IUnknown
    {
    public:
        virtual /* [helpstring] */ HRESULT
STDMETHODCALLTYPE doStockLevel(
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer) = 0;

        virtual /* [helpstring] */ HRESULT
STDMETHODCALLTYPE doNewOrder(
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer) = 0;
```

```
        virtual /* [helpstring] */ HRESULT
STDMETHODCALLTYPE doPayment(
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer) = 0;

        virtual /* [helpstring] */ HRESULT
STDMETHODCALLTYPE doOrderStatus(
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer) = 0;

        virtual /* [helpstring] */ HRESULT
STDMETHODCALLTYPE doDBInfo( void) = 0;

        virtual /* [helpstring] */ HRESULT
STDMETHODCALLTYPE doSetComplete( void) = 0;

    };

#else /* C style interface */

    typedef struct Itpcc_comVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE
*QueryInterface )(
            Itpcc_com * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void **ppvObject);

        ULONG ( STDMETHODCALLTYPE *AddRef )(
            Itpcc_com * This);

        ULONG ( STDMETHODCALLTYPE *Release )(
            Itpcc_com * This);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doStockLevel )(
            Itpcc_com * This,
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doNewOrder )(
            Itpcc_com * This,
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doPayment )(
            Itpcc_com * This,
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doOrderStatus )(
            Itpcc_com * This,
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doDBInfo )(
            Itpcc_com * This);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doSetComplete )(
            Itpcc_com * This);

        END_INTERFACE
    } Itpcc_comVtbl;

    interface Itpcc_com
    {
        CONST_VTBL struct Itpcc_comVtbl *lpVtbl;
    };



#ifdef COBJMACROS


#define Itpcc_com_QueryInterface(This,riid,ppvObject)\
    (This)->lpVtbl -> QueryInterface(This,riid,ppvObject)

#define Itpcc_com_AddRef(This)  \
    (This)->lpVtbl -> AddRef(This)

#define Itpcc_com_Release(This) \
    (This)->lpVtbl -> Release(This)


#define Itpcc_com_doStockLevel(This,size,buffer) \
    (This)->lpVtbl -> doStockLevel(This,size,buffer)

#define Itpcc_com_doNewOrder(This,size,buffer)  \
    (This)->lpVtbl -> doNewOrder(This,size,buffer)

#define Itpcc_com_doPayment(This,size,buffer) \
    (This)->lpVtbl -> doPayment(This,size,buffer)

#define Itpcc_com_doOrderStatus(This,size,buffer)  \
    (This)->lpVtbl -> doOrderStatus(This,size,buffer)

#define Itpcc_com_doDBInfo(This)  \
    (This)->lpVtbl -> doDBInfo(This)

#define Itpcc_com_doSetComplete(This)  \
    (This)->lpVtbl -> doSetComplete(This)

#endif /* COBJMACROS */


#endif   /* C style interface */



/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com_doStockLevel_Proxy(
    Itpcc_com * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);


void __RPC_STUB Itpcc_com_doStockLevel_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com_doNewOrder_Proxy(
    Itpcc_com * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);


void __RPC_STUB Itpcc_com_doNewOrder_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com_doPayment_Proxy(
    Itpcc_com * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);


void __RPC_STUB Itpcc_com_doPayment_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
```

```
    DWORD *_pdwStubPhase);


/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com_doOrderStatus_Proxy(
    Itpcc_com * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);


void __RPC_STUB Itpcc_com_doOrderStatus_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com_doDBInfo_Proxy(
    Itpcc_com * This);


void __RPC_STUB Itpcc_com_doDBInfo_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);


/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com_doSetComplete_Proxy(
    Itpcc_com * This);


void __RPC_STUB Itpcc_com_doSetComplete_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);



#endif   /* __Itpcc_com_INTERFACE_DEFINED__ */



#ifndef __tpccComLib_LIBRARY_DEFINED__
#define __tpccComLib_LIBRARY_DEFINED__

/* library tpccComLib */
/* [custom][helpstring][version][uuid] */


EXTERN_C const IID LIBID_tpccComLib;

EXTERN_C const CLSID CLSID_CompReg;

#ifdef __cplusplus

class DECLSPEC_UUID("90EEDAFF-F8D3-4711-99A9-
8AC3C0FE5DB9")
CompReg;
#endif

EXTERN_C const CLSID CLSID_tpcc_com;

#ifdef __cplusplus

class DECLSPEC_UUID("5F752BF2-F739-43D4-8492-
44C19581C0A1")
tpcc_com;
#endif
#endif /* __tpccComLib_LIBRARY_DEFINED__ */

/* Additional Prototypes for ALL interfaces */

unsigned long             __RPC_USER
BSTR_UserSize(     unsigned long *, unsigned long            ,
BSTR * );
unsigned char * __RPC_USER
BSTR_UserMarshal(  unsigned long *, unsigned char *,
BSTR * );
unsigned char * __RPC_USER
BSTR_UserUnmarshal(unsigned long *, unsigned char *,
BSTR * );
void                      __RPC_USER
BSTR_UserFree(     unsigned long *, BSTR * );

unsigned long             __RPC_USER
LPSAFEARRAY_UserSize(     unsigned long *, unsigned
long           , LPSAFEARRAY * );
unsigned char * __RPC_USER
LPSAFEARRAY_UserMarshal(  unsigned long *, unsigned
char *, LPSAFEARRAY * );
unsigned char * __RPC_USER
LPSAFEARRAY_UserUnmarshal(unsigned long *, unsigned
char *, LPSAFEARRAY * );
void                      __RPC_USER
LPSAFEARRAY_UserFree(     unsigned long *,
LPSAFEARRAY * );

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

tpccCom_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 Sat Feb 09 14:06:08 2008
 */
/* Compiler settings for .\tpccCom.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,
IID_IComponentRegistrar,0xa817e7a2,0x43fa,0x11d0,0x9e,0
x44,0x00,0xaa,0x00,0xb6,0x77,0x0a);


MIDL_DEFINE_GUID(IID,
IID_Itpcc_com,0x5B4FA473,0x2E68,0x4D79,0xA6,0x26,0xF
3,0x8B,0x30,0xB8,0x19,0x6E);


MIDL_DEFINE_GUID(IID,
LIBID_tpccComLib,0x91F1B8B0,0x89E9,0x457B,0xA2,0x28,
0x3E,0x2D,0x6C,0xE3,0xE7,0x52);
```

```
MIDL_DEFINE_GUID(CLSID,
CLSID_CompReg,0x90EEDAFF,0xF8D3,0x4711,0x99,0xA9,
0x8A,0xC3,0xC0,0xFE,0x5D,0xB9);


MIDL_DEFINE_GUID(CLSID,
CLSID_tpcc_com,0x5F752BF2,0xF739,0x43D4,0x84,0x92,0
x44,0xC1,0x95,0x81,0xC0,0xA1);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif



#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/
```

tpccCom.idl

```
// tpccCom.idl : IDL source for tpccCom
//

// This file will be processed by the MIDL tool to
// produce the type library (tpccCom.tlb) and marshalling code.

import "oaidl.idl";
import "ocidl.idl";
//this is test.

[
   object,
   uuid(a817e7a2-43fa-11d0-9e44-00aa00b6770a),
   dual,
   helpstring("IComponentRegistrar Interface"),
   pointer_default(unique)
]
interface IComponentRegistrar : IDispatch
{
   [id(1)] HRESULT Attach([in] BSTR bstrPath);
   [id(2)] HRESULT RegisterAll();
   [id(3)] HRESULT UnregisterAll();
   [id(4)] HRESULT GetComponents([out]
SAFEARRAY(BSTR)* pbstrCLSIDs, [out]
SAFEARRAY(BSTR)* pbstrDescriptions);
   [id(5)]HRESULT RegisterComponent([in] BSTR bstrCLSID);
   [id(6)] HRESULT UnregisterComponent([in] BSTR
bstrCLSID);
};
```

```
[
   object,
   uuid(5B4FA473-2E68-4D79-A626-F38B30B8196E),
   helpstring("Itpcc_com Interface"),
   pointer_default(unique)
]
interface Itpcc_com : IUnknown{
   [helpstring("method doStockLevel")] HRESULT
doStockLevel([in] INT *size, [in,out, size_is(,*size)] UCHAR
**buffer);
   [helpstring("method doNewOrder")] HRESULT
doNewOrder([in] INT* size, [in,out,size_is(,*size)] UCHAR**
buffer);
   [helpstring("method doPayment")] HRESULT
doPayment([in] INT* size, [in,out,size_is(,*size)] UCHAR**
buffer);
   [helpstring("method doOrderStatus")] HRESULT
doOrderStatus([in] INT* size, [in,out,size_is(,*size)] UCHAR**
buffer);
   [helpstring("method doDBInfo")] HRESULT doDBInfo(void);
   [helpstring("method doSetComplete")] HRESULT
doSetComplete(void);
};
[
   uuid(91F1B8B0-89E9-457B-A228-3E2D6CE3E752),
   version(1.0),
   helpstring("tpccCom 1.0 Type Library"),
   custom(a817e7a1-43fa-11d0-9e44-
00aa00b6770a,"{90EEDAFF-F8D3-4711-99A9-
8AC3C0FE5DB9}")
]
library tpccComLib
{
   importlib("stdole2.tlb");

   [
      uuid(90EEDAFF-F8D3-4711-99A9-8AC3C0FE5DB9),
      helpstring("ComponentRegistrar Class")
   ]
   coclass CompReg
   {
      [default] interface IComponentRegistrar;
   };
   [
      uuid(5F752BF2-F739-43D4-8492-44C19581C0A1),
      helpstring("tpcc_com Class")
   ]
   coclass tpcc_com
   {
      [default] interface Itpcc_com;
   };
};
```

tpccCom_p.c

```
/* this ALWAYS GENERATED file contains the proxy stub
code */


 /* File created by MIDL compiler version 6.00.0361 */
/* at Sat Feb 09 14:06:08 2008
 */
/* Compiler settings for .\tpccCom.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 */
#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__ 475
#endif


#include "rpcproxy.h"
#ifndef __RPCPROXY_H_VERSION__
#error this stub requires an updated version of <rpcproxy.h>
#endif // __RPCPROXY_H_VERSION__


#include "tpccCom.h"


#define TYPE_FORMAT_STRING_SIZE   1089
#define PROC_FORMAT_STRING_SIZE   409
#define TRANSMIT_AS_TABLE_SIZE    0
#define WIRE_MARSHAL_TABLE_SIZE   2

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
IComponentRegistrar_ServerInfo;
extern const MIDL_STUBLESS_PROXY_INFO
IComponentRegistrar_ProxyInfo;


extern const MIDL_STUB_DESC Object_StubDesc;


extern const MIDL_SERVER_INFO Itpcc_com_ServerInfo;
extern const MIDL_STUBLESS_PROXY_INFO
Itpcc_com_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_NT50_OR_LATER)
#error You need a Windows 2000 or later to run this stub
because it uses these features:
#error   /robust command line switch.
#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 Attach */

            0x33,    /* FC_AUTO_HANDLE */
            0x6c,    /* Old Flags:  object, Oi2 */
/*  2 */   NdrFcLong( 0x0 ),/* 0 */
/*  6 */   NdrFcShort( 0x7 ),   /* 7 */
/*  8 */   NdrFcShort( 0xc ),   /* x86 Stack size/offset = 12 */
/* 10 */   NdrFcShort( 0x0 ),   /* 0 */
/* 12 */   NdrFcShort( 0x8 ),   /* 8 */
/* 14 */   0x46,    /* Oi2 Flags:  clt must size, has return, has
ext, */
            0x2,     /* 2 */
/* 16 */   0x8,     /* 8 */
            0x5,     /* Ext Flags:  new corr desc, srv corr check, */
/* 18 */   NdrFcShort( 0x0 ),   /* 0 */
/* 20 */   NdrFcShort( 0x1 ),   /* 1 */
/* 22 */   NdrFcShort( 0x0 ),   /* 0 */

    /* Parameter bstrPath */

/* 24 */   NdrFcShort( 0x8b ),/* Flags:  must size, must free, in,
by val, */
/* 26 */   NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 28 */   NdrFcShort( 0x1c ), /* Type Offset=28 */

    /* Return value */

/* 30 */   NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 32 */   NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 34 */   0x8,     /* FC_LONG */
            0x0,     /* 0 */
```

```
   /* Procedure doSetComplete */


   /* Procedure RegisterAll */

/* 36 */  0x33,    /* FC_AUTO_HANDLE */
        0x6c,    /* Old Flags:  object, Oi2 */
/* 38 */  NdrFcLong( 0x0 ),/* 0 */
/* 42 */  NdrFcShort( 0x8 ),   /* 8 */
/* 44 */  NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 46 */  NdrFcShort( 0x0 ),   /* 0 */
/* 48 */  NdrFcShort( 0x8 ),   /* 8 */
/* 50 */  0x44,    /* Oi2 Flags:  has return, has ext, */
        0x1,     /* 1 */
/* 52 */  0x8,     /* 8 */
        0x1,     /* Ext Flags:  new corr desc, */
/* 54 */  NdrFcShort( 0x0 ),   /* 0 */
/* 56 */  NdrFcShort( 0x0 ),   /* 0 */
/* 58 */  NdrFcShort( 0x0 ),   /* 0 */

   /* Return value */


   /* Return value */

/* 60 */  NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 62 */  NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 64 */  0x8,     /* FC_LONG */
        0x0,     /* 0 */

   /* Procedure UnregisterAll */

/* 66 */  0x33,    /* FC_AUTO_HANDLE */
        0x6c,    /* Old Flags:  object, Oi2 */
/* 68 */  NdrFcLong( 0x0 ),/* 0 */
/* 72 */  NdrFcShort( 0x9 ),   /* 9 */
/* 74 */  NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 76 */  NdrFcShort( 0x0 ),   /* 0 */
/* 78 */  NdrFcShort( 0x8 ),   /* 8 */
/* 80 */  0x44,    /* Oi2 Flags:  has return, has ext, */
        0x1,     /* 1 */
/* 82 */  0x8,     /* 8 */
        0x1,     /* Ext Flags:  new corr desc, */
/* 84 */  NdrFcShort( 0x0 ),   /* 0 */
/* 86 */  NdrFcShort( 0x0 ),   /* 0 */
/* 88 */  NdrFcShort( 0x0 ),   /* 0 */

   /* Return value */

/* 90 */  NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 92 */  NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 94 */  0x8,     /* FC_LONG */
        0x0,     /* 0 */

   /* Procedure GetComponents */

/* 96 */  0x33,    /* FC_AUTO_HANDLE */
        0x6c,    /* Old Flags:  object, Oi2 */
/* 98 */  NdrFcLong( 0x0 ),/* 0 */
/* 102 */ NdrFcShort( 0xa ),   /* 10 */
/* 104 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 106 */ NdrFcShort( 0x0 ),   /* 0 */
/* 108 */ NdrFcShort( 0x8 ),   /* 8 */
/* 110 */ 0x45,    /* Oi2 Flags:  srv must size, has return, has
ext, */
        0x3,     /* 3 */
/* 112 */ 0x8,     /* 8 */
        0x3,     /* Ext Flags:  new corr desc, clt corr check, */
/* 114 */ NdrFcShort( 0x24 ), /* 36 */
/* 116 */ NdrFcShort( 0x0 ),   /* 0 */
/* 118 */ NdrFcShort( 0x0 ),   /* 0 */

   /* Parameter pbstrCLSIDs */

/* 120 */ NdrFcShort( 0x2113 ),/* Flags:  must size, must free,
out, simple ref, srv alloc size=8 */
/* 122 */ NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 124 */ NdrFcShort( 0x41e ),  /* Type Offset=1054 */

   /* Parameter pbstrDescriptions */

/* 126 */ NdrFcShort( 0x2113 ),/* Flags:  must size, must free,
out, simple ref, srv alloc size=8 */
/* 128 */ NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 130 */ NdrFcShort( 0x41e ),  /* Type Offset=1054 */

   /* Return value */

/* 132 */ NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 134 */ NdrFcShort( 0xc ),   /* x86 Stack size/offset = 12 */
/* 136 */ 0x8,     /* FC_LONG */
        0x0,     /* 0 */

   /* Procedure RegisterComponent */

/* 138 */ 0x33,    /* FC_AUTO_HANDLE */
        0x6c,    /* Old Flags:  object, Oi2 */
/* 140 */ NdrFcLong( 0x0 ),/* 0 */
/* 144 */ NdrFcShort( 0xb ),   /* 11 */
/* 146 */ NdrFcShort( 0xc ),   /* x86 Stack size/offset = 12 */
/* 148 */ NdrFcShort( 0x0 ),   /* 0 */
/* 150 */ NdrFcShort( 0x8 ),   /* 8 */
/* 152 */ 0x46,    /* Oi2 Flags:  clt must size, has return, has
ext, */
        0x2,     /* 2 */
/* 154 */ 0x8,     /* 8 */
        0x5,     /* Ext Flags:  new corr desc, srv corr check, */
/* 156 */ NdrFcShort( 0x0 ),   /* 0 */
/* 158 */ NdrFcShort( 0x1 ),   /* 1 */
/* 160 */ NdrFcShort( 0x0 ),   /* 0 */

   /* Parameter bstrCLSID */

/* 162 */ NdrFcShort( 0x8b ),/* Flags:  must size, must free, in,
by val, */
/* 164 */ NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 166 */ NdrFcShort( 0x1c ), /* Type Offset=28 */

   /* Return value */

/* 168 */ NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 170 */ NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 172 */ 0x8,     /* FC_LONG */
        0x0,     /* 0 */

   /* Procedure UnregisterComponent */

/* 174 */ 0x33,    /* FC_AUTO_HANDLE */
        0x6c,    /* Old Flags:  object, Oi2 */
/* 176 */ NdrFcLong( 0x0 ),/* 0 */
/* 180 */ NdrFcShort( 0xc ),   /* 12 */
/* 182 */ NdrFcShort( 0xc ),   /* x86 Stack size/offset = 12 */
/* 184 */ NdrFcShort( 0x0 ),   /* 0 */
/* 186 */ NdrFcShort( 0x8 ),   /* 8 */
/* 188 */ 0x46,    /* Oi2 Flags:  clt must size, has return, has
ext, */
        0x2,     /* 2 */
/* 190 */ 0x8,     /* 8 */
        0x5,     /* Ext Flags:  new corr desc, srv corr check, */
/* 192 */ NdrFcShort( 0x0 ),   /* 0 */
/* 194 */ NdrFcShort( 0x1 ),   /* 1 */
/* 196 */ NdrFcShort( 0x0 ),   /* 0 */

   /* Parameter bstrCLSID */

/* 198 */ NdrFcShort( 0x8b ),/* Flags:  must size, must free, in,
by val, */
/* 200 */ NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 202 */ NdrFcShort( 0x1c ), /* Type Offset=28 */

   /* Return value */
```

```
/* 204 */NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 206 */NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 208 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Procedure doStockLevel */

/* 210 */0x33,    /* FC_AUTO_HANDLE */
        0x6c,     /* Old Flags:  object, Oi2 */
/* 212 */NdrFcLong( 0x0 ),/* 0 */
/* 216 */NdrFcShort( 0x3 ),   /* 3 */
/* 218 */NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 220 */NdrFcShort( 0x1c ), /* 28 */
/* 222 */NdrFcShort( 0x8 ),   /* 8 */
/* 224 */0x47,    /* Oi2 Flags:  srv must size, clt must size,
has return, has ext, */
        0x3,      /* 3 */
/* 226 */0x8,      /* 8 */
        0x7,      /* Ext Flags:  new corr desc, clt corr check, srv
corr check, */
/* 228 */NdrFcShort( 0x1 ),   /* 1 */
/* 230 */NdrFcShort( 0x1 ),   /* 1 */
/* 232 */NdrFcShort( 0x0 ),   /* 0 */

   /* Parameter size */

/* 234 */NdrFcShort( 0x148 ),  /* Flags:  in, base type, simple
ref, */
/* 236 */NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 238 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Parameter buffer */

/* 240 */NdrFcShort( 0x201b ),/* Flags:  must size, must free,
in, out, srv alloc size=8 */
/* 242 */NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 244 */NdrFcShort( 0x42c ),  /* Type Offset=1068 */

   /* Return value */

/* 246 */NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 248 */NdrFcShort( 0xc ),   /* x86 Stack size/offset = 12 */
/* 250 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Procedure doNewOrder */

/* 252 */0x33,    /* FC_AUTO_HANDLE */
        0x6c,     /* Old Flags:  object, Oi2 */
/* 254 */NdrFcLong( 0x0 ),/* 0 */
/* 258 */NdrFcShort( 0x4 ),   /* 4 */
/* 260 */NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 262 */NdrFcShort( 0x1c ), /* 28 */
/* 264 */NdrFcShort( 0x8 ),   /* 8 */
/* 266 */0x47,    /* Oi2 Flags:  srv must size, clt must size,
has return, has ext, */
        0x3,      /* 3 */
/* 268 */0x8,      /* 8 */
        0x7,      /* Ext Flags:  new corr desc, clt corr check, srv
corr check, */
/* 270 */NdrFcShort( 0x1 ),   /* 1 */
/* 272 */NdrFcShort( 0x1 ),   /* 1 */
/* 274 */NdrFcShort( 0x0 ),   /* 0 */

   /* Parameter size */

/* 276 */NdrFcShort( 0x148 ),  /* Flags:  in, base type, simple
ref, */
/* 278 */NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 280 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Parameter buffer */

/* 282 */NdrFcShort( 0x201b ),/* Flags:  must size, must free,
in, out, srv alloc size=8 */
/* 284 */NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 286 */NdrFcShort( 0x42c ),  /* Type Offset=1068 */

   /* Return value */

/* 288 */NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 290 */NdrFcShort( 0xc ),   /* x86 Stack size/offset = 12 */
/* 292 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Procedure doPayment */

/* 294 */0x33,    /* FC_AUTO_HANDLE */
        0x6c,     /* Old Flags:  object, Oi2 */
/* 296 */NdrFcLong( 0x0 ),/* 0 */
/* 300 */NdrFcShort( 0x5 ),   /* 5 */
/* 302 */NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 304 */NdrFcShort( 0x1c ), /* 28 */
/* 306 */NdrFcShort( 0x8 ),   /* 8 */
/* 308 */0x47,    /* Oi2 Flags:  srv must size, clt must size,
has return, has ext, */
        0x3,      /* 3 */
/* 310 */0x8,      /* 8 */
        0x7,      /* Ext Flags:  new corr desc, clt corr check, srv
corr check, */
/* 312 */NdrFcShort( 0x1 ),   /* 1 */
/* 314 */NdrFcShort( 0x1 ),   /* 1 */
/* 316 */NdrFcShort( 0x0 ),   /* 0 */

   /* Parameter size */

/* 318 */NdrFcShort( 0x148 ),  /* Flags:  in, base type, simple
ref, */
/* 320 */NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 322 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Parameter buffer */

/* 324 */NdrFcShort( 0x201b ),/* Flags:  must size, must free,
in, out, srv alloc size=8 */
/* 326 */NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 328 */NdrFcShort( 0x42c ),  /* Type Offset=1068 */

   /* Return value */

/* 330 */NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 332 */NdrFcShort( 0xc ),   /* x86 Stack size/offset = 12 */
/* 334 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Procedure doOrderStatus */

/* 336 */0x33,    /* FC_AUTO_HANDLE */
        0x6c,     /* Old Flags:  object, Oi2 */
/* 338 */NdrFcLong( 0x0 ),/* 0 */
/* 342 */NdrFcShort( 0x6 ),   /* 6 */
/* 344 */NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 346 */NdrFcShort( 0x1c ), /* 28 */
/* 348 */NdrFcShort( 0x8 ),   /* 8 */
/* 350 */0x47,    /* Oi2 Flags:  srv must size, clt must size,
has return, has ext, */
        0x3,      /* 3 */
/* 352 */0x8,      /* 8 */
        0x7,      /* Ext Flags:  new corr desc, clt corr check, srv
corr check, */
/* 354 */NdrFcShort( 0x1 ),   /* 1 */
/* 356 */NdrFcShort( 0x1 ),   /* 1 */
/* 358 */NdrFcShort( 0x0 ),   /* 0 */

   /* Parameter size */

/* 360 */NdrFcShort( 0x148 ),  /* Flags:  in, base type, simple
ref, */
```

```
/* 362 */NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 364 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Parameter buffer */

/* 366 */NdrFcShort( 0x201b ),/* Flags:  must size, must free,
in, out, srv alloc size=8 */
/* 368 */NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 370 */NdrFcShort( 0x42c ),  /* Type Offset=1068 */

   /* Return value */

/* 372 */NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 374 */NdrFcShort( 0xc ),   /* x86 Stack size/offset = 12 */
/* 376 */0x8,      /* FC_LONG */
        0x0,      /* 0 */

   /* Procedure doDBInfo */

/* 378 */0x33,    /* FC_AUTO_HANDLE */
        0x6c,    /* Old Flags:  object, Oi2 */
/* 380 */NdrFcLong( 0x0 ),/* 0 */
/* 384 */NdrFcShort( 0x7 ),   /* 7 */
/* 386 */NdrFcShort( 0x8 ),   /* x86 Stack size/offset = 8 */
/* 388 */NdrFcShort( 0x0 ),   /* 0 */
/* 390 */NdrFcShort( 0x8 ),   /* 8 */
/* 392 */0x44,    /* Oi2 Flags:  has return, has ext, */
        0x1,      /* 1 */
/* 394 */0x8,      /* 8 */
        0x1,      /* Ext Flags:  new corr desc, */
/* 396 */NdrFcShort( 0x0 ),   /* 0 */
/* 398 */NdrFcShort( 0x0 ),   /* 0 */
/* 400 */NdrFcShort( 0x0 ),   /* 0 */

   /* Return value */

/* 402 */NdrFcShort( 0x70 ), /* Flags:  out, return, base type,
*/
/* 404 */NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 406 */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( 0xe ),   /* Offset= 14 (18) */
/*  6 */
        0x1b,    /* FC_CARRAY */
        0x1,      /* 1 */
/*  8 */   NdrFcShort( 0x2 ),   /* 2 */
/* 10 */  0x9,      /* Corr desc: FC_ULONG */
        0x0,      /*  */
/* 12 */  NdrFcShort( 0xfffc ),   /* -4 */
/* 14 */  NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 16 */  0x6,      /* FC_SHORT */
        0x5b,    /* FC_END */
/* 18 */
        0x17,    /* FC_CSTRUCT */
        0x3,      /* 3 */
/* 20 */  NdrFcShort( 0x8 ),   /* 8 */
/* 22 */  NdrFcShort( 0xfff0 ),   /* Offset= -16 (6) */
/* 24 */  0x8,      /* FC_LONG */
        0x8,      /* FC_LONG */
/* 26 */  0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 28 */  0xb4,     /* FC_USER_MARSHAL */
        0x83,    /* 131 */
/* 30 */  NdrFcShort( 0x0 ),   /* 0 */
/* 32 */  NdrFcShort( 0x4 ),   /* 4 */
/* 34 */  NdrFcShort( 0x0 ),   /* 0 */
/* 36 */  NdrFcShort( 0xffde ),  /* Offset= -34 (2) */
/* 38 */
        0x11, 0x4,   /* FC_RP [alloced_on_stack] */
/* 40 */  NdrFcShort( 0x3f6 ),   /* Offset= 1014 (1054) */
/* 42 */
        0x13, 0x10, /* FC_OP [pointer_deref] */
/* 44 */  NdrFcShort( 0x2 ),   /* Offset= 2 (46) */
/* 46 */
        0x13, 0x0,   /* FC_OP */
/* 48 */  NdrFcShort( 0x3dc ),  /* Offset= 988 (1036) */
/* 50 */
        0x2a,    /* FC_ENCAPSULATED_UNION */
        0x49,    /* 73 */
/* 52 */  NdrFcShort( 0x18 ), /* 24 */
/* 54 */  NdrFcShort( 0xa ),   /* 10 */
/* 56 */  NdrFcLong( 0x8 ),/* 8 */
/* 60 */  NdrFcShort( 0x5a ), /* Offset= 90 (150) */
/* 62 */  NdrFcLong( 0xd ),/* 13 */
/* 66 */  NdrFcShort( 0x90 ), /* Offset= 144 (210) */
/* 68 */  NdrFcLong( 0x9 ),/* 9 */
/* 72 */  NdrFcShort( 0xc2 ), /* Offset= 194 (266) */
/* 74 */  NdrFcLong( 0xc ),/* 12 */
/* 78 */  NdrFcShort( 0x2c0 ),  /* Offset= 704 (782) */
/* 80 */  NdrFcLong( 0x24 ),  /* 36 */
/* 84 */  NdrFcShort( 0x2ea ),  /* Offset= 746 (830) */
/* 86 */  NdrFcLong( 0x800d ),/* 32781 */
/* 90 */  NdrFcShort( 0x306 ),  /* Offset= 774 (864) */
/* 92 */  NdrFcLong( 0x10 ),  /* 16 */
/* 96 */  NdrFcShort( 0x320 ),  /* Offset= 800 (896) */
/* 98 */  NdrFcLong( 0x2 ),/* 2 */
/* 102 */NdrFcShort( 0x33a ),  /* Offset= 826 (928) */
/* 104 */NdrFcLong( 0x3 ),/* 3 */
/* 108 */NdrFcShort( 0x354 ),  /* Offset= 852 (960) */
/* 110 */NdrFcLong( 0x14 ),  /* 20 */
/* 114 */NdrFcShort( 0x36e ),  /* Offset= 878 (992) */
/* 116 */NdrFcShort( 0xffff ),/* Offset= -1 (115) */
/* 118 */
        0x1b,    /* FC_CARRAY */
        0x3,      /* 3 */
/* 120 */NdrFcShort( 0x4 ),   /* 4 */
/* 122 */0x19,    /* Corr desc:  field pointer, FC_ULONG */
        0x0,      /*  */
/* 124 */NdrFcShort( 0x0 ),   /* 0 */
/* 126 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 128 */
        0x4b,    /* FC_PP */
        0x5c,    /* FC_PAD */
/* 130 */
        0x48,    /* FC_VARIABLE_REPEAT */
        0x49,    /* FC_FIXED_OFFSET */
/* 132 */NdrFcShort( 0x4 ),   /* 4 */
/* 134 */NdrFcShort( 0x0 ),   /* 0 */
/* 136 */NdrFcShort( 0x1 ),   /* 1 */
/* 138 */NdrFcShort( 0x0 ),   /* 0 */
/* 140 */NdrFcShort( 0x0 ),   /* 0 */
/* 142 */0x13, 0x0,   /* FC_OP */
/* 144 */NdrFcShort( 0xff82 ),  /* Offset= -126 (18) */
/* 146 */
        0x5b,    /* FC_END */

        0x8,      /* FC_LONG */
/* 148 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 150 */
        0x16,    /* FC_PSTRUCT */
        0x3,      /* 3 */
/* 152 */NdrFcShort( 0x8 ),   /* 8 */
/* 154 */
        0x4b,    /* FC_PP */
        0x5c,    /* FC_PAD */
/* 156 */
        0x46,    /* FC_NO_REPEAT */
        0x5c,    /* FC_PAD */
/* 158 */NdrFcShort( 0x4 ),   /* 4 */
/* 160 */NdrFcShort( 0x4 ),   /* 4 */
```

```
/* 162 */0x11, 0x0,   /* FC_RP */
/* 164 */NdrFcShort( 0xffd2 ),  /* Offset= -46 (118) */
/* 166 */
        0x5b,    /* FC_END */

        0x8,     /* FC_LONG */
/* 168 */0x8,       /* FC_LONG */
        0x5b,    /* FC_END */
/* 170 */
        0x2f,    /* FC_IP */
        0x5a,    /* FC_CONSTANT_IID */
/* 172 */NdrFcLong( 0x0 ),/* 0 */
/* 176 */NdrFcShort( 0x0 ),   /* 0 */
/* 178 */NdrFcShort( 0x0 ),   /* 0 */
/* 180 */0xc0,     /* 192 */
        0x0,     /* 0 */
/* 182 */0x0,       /* 0 */
        0x0,     /* 0 */
/* 184 */0x0,       /* 0 */
        0x0,     /* 0 */
/* 186 */0x0,       /* 0 */
        0x46,    /* 70 */
/* 188 */
        0x21,    /* FC_BOGUS_ARRAY */
        0x3,     /* 3 */
/* 190 */NdrFcShort( 0x0 ),   /* 0 */
/* 192 */0x19,    /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 194 */NdrFcShort( 0x0 ),   /* 0 */
/* 196 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 198 */NdrFcLong( 0xffffffff ),/* -1 */
/* 202 */NdrFcShort( 0x0 ),   /* Corr flags:  */
/* 204 */0x4c,     /* FC_EMBEDDED_COMPLEX */
        0x0,     /* 0 */
/* 206 */NdrFcShort( 0xffdc ),  /* Offset= -36 (170) */
/* 208 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 210 */
        0x1a,    /* FC_BOGUS_STRUCT */
        0x3,     /* 3 */
/* 212 */NdrFcShort( 0x8 ),   /* 8 */
/* 214 */NdrFcShort( 0x0 ),   /* 0 */
/* 216 */NdrFcShort( 0x6 ),   /* Offset= 6 (222) */
/* 218 */0x8,       /* FC_LONG */
        0x36,    /* FC_POINTER */
/* 220 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 222 */
        0x11, 0x0,   /* FC_RP */
/* 224 */NdrFcShort( 0xffdc ),  /* Offset= -36 (188) */
/* 226 */
        0x2f,    /* FC_IP */
        0x5a,    /* FC_CONSTANT_IID */
/* 228 */NdrFcLong( 0x20400 ),  /* 132096 */
/* 232 */NdrFcShort( 0x0 ),   /* 0 */
/* 234 */NdrFcShort( 0x0 ),   /* 0 */
/* 236 */0xc0,     /* 192 */
        0x0,     /* 0 */
/* 238 */0x0,       /* 0 */
        0x0,     /* 0 */
/* 240 */0x0,       /* 0 */
        0x0,     /* 0 */
/* 242 */0x0,       /* 0 */
        0x46,    /* 70 */
/* 244 */
        0x21,    /* FC_BOGUS_ARRAY */
        0x3,     /* 3 */
/* 246 */NdrFcShort( 0x0 ),   /* 0 */
/* 248 */0x19,    /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 250 */NdrFcShort( 0x0 ),   /* 0 */
/* 252 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 254 */NdrFcLong( 0xffffffff ),/* -1 */
/* 258 */NdrFcShort( 0x0 ),   /* Corr flags:  */
/* 260 */0x4c,     /* FC_EMBEDDED_COMPLEX */
        0x0,     /* 0 */
/* 262 */NdrFcShort( 0xffdc ),  /* Offset= -36 (226) */
/* 264 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 266 */
        0x1a,    /* FC_BOGUS_STRUCT */
        0x3,     /* 3 */
/* 268 */NdrFcShort( 0x8 ),   /* 8 */
/* 270 */NdrFcShort( 0x0 ),   /* 0 */
/* 272 */NdrFcShort( 0x6 ),   /* Offset= 6 (278) */
/* 274 */0x8,       /* FC_LONG */
        0x36,    /* FC_POINTER */
/* 276 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 278 */
        0x11, 0x0,   /* FC_RP */
/* 280 */NdrFcShort( 0xffdc ),  /* Offset= -36 (244) */
/* 282 */
        0x2b,    /* FC_NON_ENCAPSULATED_UNION */
        0x9,     /* FC_ULONG */
/* 284 */0x7,       /* Corr desc: FC_USHORT */
        0x0,     /*  */
/* 286 */NdrFcShort( 0xfff8 ),   /* -8 */
/* 288 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 290 */NdrFcShort( 0x2 ),   /* Offset= 2 (292) */
/* 292 */NdrFcShort( 0x10 ),/* 16 */
/* 294 */NdrFcShort( 0x2f ),  /* 47 */
/* 296 */NdrFcLong( 0x14 ),  /* 20 */
/* 300 */NdrFcShort( 0x800b ),/* Simple arm type:
FC_HYPER */
/* 302 */NdrFcLong( 0x3 ),/* 3 */
/* 306 */NdrFcShort( 0x8008 ),/* Simple arm type: FC_LONG
*/
/* 308 */NdrFcLong( 0x11 ),  /* 17 */
/* 312 */NdrFcShort( 0x8001 ),/* Simple arm type: FC_BYTE
*/
/* 314 */NdrFcLong( 0x2 ),/* 2 */
/* 318 */NdrFcShort( 0x8006 ),/* Simple arm type:
FC_SHORT */
/* 320 */NdrFcLong( 0x4 ),/* 4 */
/* 324 */NdrFcShort( 0x800a ),/* Simple arm type:
FC_FLOAT */
/* 326 */NdrFcLong( 0x5 ),/* 5 */
/* 330 */NdrFcShort( 0x800c ),/* Simple arm type:
FC_DOUBLE */
/* 332 */NdrFcLong( 0xb ),/* 11 */
/* 336 */NdrFcShort( 0x8006 ),/* Simple arm type:
FC_SHORT */
/* 338 */NdrFcLong( 0xa ),/* 10 */
/* 342 */NdrFcShort( 0x8008 ),/* Simple arm type: FC_LONG
*/
/* 344 */NdrFcLong( 0x6 ),/* 6 */
/* 348 */NdrFcShort( 0xe8 ),/* Offset= 232 (580) */
/* 350 */NdrFcLong( 0x7 ),/* 7 */
/* 354 */NdrFcShort( 0x800c ),/* Simple arm type:
FC_DOUBLE */
/* 356 */NdrFcLong( 0x8 ),/* 8 */
/* 360 */NdrFcShort( 0xe2 ),/* Offset= 226 (586) */
/* 362 */NdrFcLong( 0xd ),/* 13 */
/* 366 */NdrFcShort( 0xff3c ),  /* Offset= -196 (170) */
/* 368 */NdrFcLong( 0x9 ),/* 9 */
/* 372 */NdrFcShort( 0xff6e ),  /* Offset= -146 (226) */
/* 374 */NdrFcLong( 0x2000 ),/* 8192 */
/* 378 */NdrFcShort( 0xd4 ),/* Offset= 212 (590) */
/* 380 */NdrFcLong( 0x24 ),  /* 36 */
/* 384 */NdrFcShort( 0xd6 ),/* Offset= 214 (598) */
/* 386 */NdrFcLong( 0x4024 ),/* 16420 */
/* 390 */NdrFcShort( 0xd0 ),/* Offset= 208 (598) */
/* 392 */NdrFcLong( 0x4011 ),/* 16401 */
/* 396 */NdrFcShort( 0x100 ),  /* Offset= 256 (652) */
/* 398 */NdrFcLong( 0x4002 ),/* 16386 */
/* 402 */NdrFcShort( 0xfe ),  /* Offset= 254 (656) */
/* 404 */NdrFcLong( 0x4003 ),/* 16387 */
/* 408 */NdrFcShort( 0xfc ),  /* Offset= 252 (660) */
/* 410 */NdrFcLong( 0x4014 ),/* 16404 */
/* 414 */NdrFcShort( 0xfa ),  /* Offset= 250 (664) */
/* 416 */NdrFcLong( 0x4004 ),/* 16388 */
/* 420 */NdrFcShort( 0xf8 ),  /* Offset= 248 (668) */
/* 422 */NdrFcLong( 0x4005 ),/* 16389 */
/* 426 */NdrFcShort( 0xf6 ),  /* Offset= 246 (672) */
```

```
/* 428 */NdrFcLong( 0x400b ), /* 16395 */
/* 432 */NdrFcShort( 0xe0 ), /* Offset= 224 (656) */
/* 434 */NdrFcLong( 0x400a ), /* 16394 */
/* 438 */NdrFcShort( 0xde ), /* Offset= 222 (660) */
/* 440 */NdrFcLong( 0x4006 ), /* 16390 */
/* 444 */NdrFcShort( 0xe8 ), /* Offset= 232 (676) */
/* 446 */NdrFcLong( 0x4007 ), /* 16391 */
/* 450 */NdrFcShort( 0xde ), /* Offset= 222 (672) */
/* 452 */NdrFcLong( 0x4008 ), /* 16392 */
/* 456 */NdrFcShort( 0xe0 ), /* Offset= 224 (680) */
/* 458 */NdrFcLong( 0x400d ), /* 16397 */
/* 462 */NdrFcShort( 0xde ), /* Offset= 222 (684) */
/* 464 */NdrFcLong( 0x4009 ), /* 16393 */
/* 468 */NdrFcShort( 0xdc ), /* Offset= 220 (688) */
/* 470 */NdrFcLong( 0x6000 ), /* 24576 */
/* 474 */NdrFcShort( 0xda ), /* Offset= 218 (692) */
/* 476 */NdrFcLong( 0x400c ), /* 16396 */
/* 480 */NdrFcShort( 0xe0 ), /* Offset= 224 (704) */
/* 482 */NdrFcLong( 0x10 ), /* 16 */
/* 486 */NdrFcShort( 0x8002 ),/* Simple arm type: FC_CHAR
*/
/* 488 */NdrFcLong( 0x12 ), /* 18 */
/* 492 */NdrFcShort( 0x8006 ),/* Simple arm type:
FC_SHORT */
/* 494 */NdrFcLong( 0x13 ), /* 19 */
/* 498 */NdrFcShort( 0x8008 ),/* Simple arm type: FC_LONG
*/
/* 500 */NdrFcLong( 0x15 ), /* 21 */
/* 504 */NdrFcShort( 0x800b ),/* Simple arm type:
FC_HYPER */
/* 506 */NdrFcLong( 0x16 ), /* 22 */
/* 510 */NdrFcShort( 0x8008 ),/* Simple arm type: FC_LONG
*/
/* 512 */NdrFcLong( 0x17 ), /* 23 */
/* 516 */NdrFcShort( 0x8008 ),/* Simple arm type: FC_LONG
*/
/* 518 */NdrFcLong( 0xe ),/* 14 */
/* 522 */NdrFcShort( 0xbe ), /* Offset= 190 (712) */
/* 524 */NdrFcLong( 0x400e ), /* 16398 */
/* 528 */NdrFcShort( 0xc2 ), /* Offset= 194 (722) */
/* 530 */NdrFcLong( 0x4010 ), /* 16400 */
/* 534 */NdrFcShort( 0xc0 ), /* Offset= 192 (726) */
/* 536 */NdrFcLong( 0x4012 ), /* 16402 */
/* 540 */NdrFcShort( 0x74 ), /* Offset= 116 (656) */
/* 542 */NdrFcLong( 0x4013 ), /* 16403 */
/* 546 */NdrFcShort( 0x72 ), /* Offset= 114 (660) */
/* 548 */NdrFcLong( 0x4015 ), /* 16405 */
/* 552 */NdrFcShort( 0x70 ), /* Offset= 112 (664) */
/* 554 */NdrFcLong( 0x4016 ), /* 16406 */
/* 558 */NdrFcShort( 0x66 ), /* Offset= 102 (660) */
/* 560 */NdrFcLong( 0x4017 ), /* 16407 */
/* 564 */NdrFcShort( 0x60 ), /* Offset= 96 (660) */
/* 566 */NdrFcLong( 0x0 ),/* 0 */
/* 570 */NdrFcShort( 0x0 ),   /* Offset= 0 (570) */
/* 572 */NdrFcLong( 0x1 ),/* 1 */
/* 576 */NdrFcShort( 0x0 ),   /* Offset= 0 (576) */
/* 578 */NdrFcShort( 0xffff ), /* Offset= -1 (577) */
/* 580 */
        0x15,    /* FC_STRUCT */
        0x7,     /* 7 */
/* 582 */NdrFcShort( 0x8 ),   /* 8 */
/* 584 */0xb,       /* FC_HYPER */
        0x5b,    /* FC_END */
/* 586 */
        0x13, 0x0,   /* FC_OP */
/* 588 */NdrFcShort( 0xfdc6 ), /* Offset= -570 (18) */
/* 590 */
        0x13, 0x10, /* FC_OP [pointer_deref] */
/* 592 */NdrFcShort( 0x2 ),   /* Offset= 2 (594) */
/* 594 */
        0x13, 0x0,   /* FC_OP */
/* 596 */NdrFcShort( 0x1b8 ),  /* Offset= 440 (1036) */
/* 598 */
        0x13, 0x0,   /* FC_OP */
/* 600 */NdrFcShort( 0x20 ), /* Offset= 32 (632) */
/* 602 */
        0x2f,    /* FC_IP */
        0x5a,    /* FC_CONSTANT_IID */
/* 604 */NdrFcLong( 0x2f ),   /* 47 */
/* 608 */NdrFcShort( 0x0 ),   /* 0 */
/* 610 */NdrFcShort( 0x0 ),   /* 0 */
/* 612 */0xc0,      /* 192 */
        0x0,     /* 0 */
/* 614 */0x0,       /* 0 */
        0x0,     /* 0 */
/* 616 */0x0,       /* 0 */
        0x0,     /* 0 */
/* 618 */0x0,       /* 0 */
        0x46,    /* 70 */
/* 620 */
        0x1b,    /* FC_CARRAY */
        0x0,     /* 0 */
/* 622 */NdrFcShort( 0x1 ),   /* 1 */
/* 624 */0x19,      /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 626 */NdrFcShort( 0x4 ),   /* 4 */
/* 628 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 630 */0x1,       /* FC_BYTE */
        0x5b,    /* FC_END */
/* 632 */
        0x1a,    /* FC_BOGUS_STRUCT */
        0x3,     /* 3 */
/* 634 */NdrFcShort( 0x10 ), /* 16 */
/* 636 */NdrFcShort( 0x0 ),   /* 0 */
/* 638 */NdrFcShort( 0xa ),   /* Offset= 10 (648) */
/* 640 */0x8,       /* FC_LONG */
        0x8,     /* FC_LONG */
/* 642 */0x4c,      /* FC_EMBEDDED_COMPLEX */
        0x0,     /* 0 */
/* 644 */NdrFcShort( 0xffd6 ),  /* Offset= -42 (602) */
/* 646 */0x36,      /* FC_POINTER */
        0x5b,    /* FC_END */
/* 648 */
        0x13, 0x0,   /* FC_OP */
/* 650 */NdrFcShort( 0xffe2 ),  /* Offset= -30 (620) */
/* 652 */
        0x13, 0x8,   /* FC_OP [simple_pointer] */
/* 654 */0x1,       /* FC_BYTE */
        0x5c,    /* FC_PAD */
/* 656 */
        0x13, 0x8,   /* FC_OP [simple_pointer] */
/* 658 */0x6,       /* FC_SHORT */
        0x5c,    /* FC_PAD */
/* 660 */
        0x13, 0x8,   /* FC_OP [simple_pointer] */
/* 662 */0x8,       /* FC_LONG */
        0x5c,    /* FC_PAD */
/* 664 */
        0x13, 0x8,   /* FC_OP [simple_pointer] */
/* 666 */0xb,       /* FC_HYPER */
        0x5c,    /* FC_PAD */
/* 668 */
        0x13, 0x8,   /* FC_OP [simple_pointer] */
/* 670 */0xa,       /* FC_FLOAT */
        0x5c,    /* FC_PAD */
/* 672 */
        0x13, 0x8,   /* FC_OP [simple_pointer] */
/* 674 */0xc,       /* FC_DOUBLE */
        0x5c,    /* FC_PAD */
/* 676 */
        0x13, 0x0,   /* FC_OP */
/* 678 */NdrFcShort( 0xff9e ),  /* Offset= -98 (580) */
/* 680 */
        0x13, 0x10, /* FC_OP [pointer_deref] */
/* 682 */NdrFcShort( 0xffa0 ),  /* Offset= -96 (586) */
/* 684 */
        0x13, 0x10, /* FC_OP [pointer_deref] */
/* 686 */NdrFcShort( 0xfdfc ),  /* Offset= -516 (170) */
/* 688 */
        0x13, 0x10, /* FC_OP [pointer_deref] */
/* 690 */NdrFcShort( 0xfe30 ), /* Offset= -464 (226) */
/* 692 */
        0x13, 0x10, /* FC_OP [pointer_deref] */
/* 694 */NdrFcShort( 0x2 ),   /* Offset= 2 (696) */
/* 696 */
        0x13, 0x10, /* FC_OP [pointer_deref] */
```

```
/* 698 */NdrFcShort( 0x2 ),   /* Offset= 2 (700) */
/* 700 */
        0x13, 0x0,   /* FC_OP */
/* 702 */NdrFcShort( 0x14e ),  /* Offset= 334 (1036) */
/* 704 */
        0x13, 0x10, /* FC_OP [pointer_deref] */
/* 706 */NdrFcShort( 0x2 ),   /* Offset= 2 (708) */
/* 708 */
        0x13, 0x0,   /* FC_OP */
/* 710 */NdrFcShort( 0x14 ), /* Offset= 20 (730) */
/* 712 */
        0x15,    /* FC_STRUCT */
        0x7,     /* 7 */
/* 714 */NdrFcShort( 0x10 ), /* 16 */
/* 716 */0x6,      /* FC_SHORT */
        0x1,     /* FC_BYTE */
/* 718 */0x1,      /* FC_BYTE */
        0x8,     /* FC_LONG */
/* 720 */0xb,      /* FC_HYPER */
        0x5b,    /* FC_END */
/* 722 */
        0x13, 0x0,   /* FC_OP */
/* 724 */NdrFcShort( 0xfff4 ),   /* Offset= -12 (712) */
/* 726 */
        0x13, 0x8,   /* FC_OP [simple_pointer] */
/* 728 */0x2,      /* FC_CHAR */
        0x5c,    /* FC_PAD */
/* 730 */
        0x1a,    /* FC_BOGUS_STRUCT */
        0x7,     /* 7 */
/* 732 */NdrFcShort( 0x20 ), /* 32 */
/* 734 */NdrFcShort( 0x0 ),   /* 0 */
/* 736 */NdrFcShort( 0x0 ),   /* Offset= 0 (736) */
/* 738 */0x8,      /* FC_LONG */
        0x8,     /* FC_LONG */
/* 740 */0x6,      /* FC_SHORT */
        0x6,     /* FC_SHORT */
/* 742 */0x6,      /* FC_SHORT */
        0x6,     /* FC_SHORT */
/* 744 */0x4c,     /* FC_EMBEDDED_COMPLEX */
        0x0,     /* 0 */
/* 746 */NdrFcShort( 0xfe30 ), /* Offset= -464 (282) */
/* 748 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 750 */
        0x1b,    /* FC_CARRAY */
        0x3,     /* 3 */
/* 752 */NdrFcShort( 0x4 ),   /* 4 */
/* 754 */0x19,     /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 756 */NdrFcShort( 0x0 ),   /* 0 */
/* 758 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 760 */
        0x4b,    /* FC_PP */
        0x5c,    /* FC_PAD */
/* 762 */
        0x48,    /* FC_VARIABLE_REPEAT */
        0x49,    /* FC_FIXED_OFFSET */
/* 764 */NdrFcShort( 0x4 ),   /* 4 */
/* 766 */NdrFcShort( 0x0 ),   /* 0 */
/* 768 */NdrFcShort( 0x1 ),   /* 1 */
/* 770 */NdrFcShort( 0x0 ),   /* 0 */
/* 772 */NdrFcShort( 0x0 ),   /* 0 */
/* 774 */0x13, 0x0,   /* FC_OP */
/* 776 */NdrFcShort( 0xffd2 ),  /* Offset= -46 (730) */
/* 778 */
        0x5b,    /* FC_END */

        0x8,     /* FC_LONG */
/* 780 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 782 */
        0x1a,    /* FC_BOGUS_STRUCT */
        0x3,     /* 3 */
/* 784 */NdrFcShort( 0x8 ),   /* 8 */
/* 786 */NdrFcShort( 0x0 ),   /* 0 */
/* 788 */NdrFcShort( 0x6 ),   /* Offset= 6 (794) */
/* 790 */0x8,      /* FC_LONG */
        0x36,    /* FC_POINTER */
/* 792 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 794 */
        0x11, 0x0,   /* FC_RP */
/* 796 */NdrFcShort( 0xffd2 ),  /* Offset= -46 (750) */
/* 798 */
        0x1b,    /* FC_CARRAY */
        0x3,     /* 3 */
/* 800 */NdrFcShort( 0x4 ),   /* 4 */
/* 802 */0x19,     /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 804 */NdrFcShort( 0x0 ),   /* 0 */
/* 806 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 808 */
        0x4b,    /* FC_PP */
        0x5c,    /* FC_PAD */
/* 810 */
        0x48,    /* FC_VARIABLE_REPEAT */
        0x49,    /* FC_FIXED_OFFSET */
/* 812 */NdrFcShort( 0x4 ),   /* 4 */
/* 814 */NdrFcShort( 0x0 ),   /* 0 */
/* 816 */NdrFcShort( 0x1 ),   /* 1 */
/* 818 */NdrFcShort( 0x0 ),   /* 0 */
/* 820 */NdrFcShort( 0x0 ),   /* 0 */
/* 822 */0x13, 0x0,   /* FC_OP */
/* 824 */NdrFcShort( 0xff40 ),  /* Offset= -192 (632) */
/* 826 */
        0x5b,    /* FC_END */

        0x8,     /* FC_LONG */
/* 828 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 830 */
        0x1a,    /* FC_BOGUS_STRUCT */
        0x3,     /* 3 */
/* 832 */NdrFcShort( 0x8 ),   /* 8 */
/* 834 */NdrFcShort( 0x0 ),   /* 0 */
/* 836 */NdrFcShort( 0x6 ),   /* Offset= 6 (842) */
/* 838 */0x8,      /* FC_LONG */
        0x36,    /* FC_POINTER */
/* 840 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 842 */
        0x11, 0x0,   /* FC_RP */
/* 844 */NdrFcShort( 0xffd2 ),  /* Offset= -46 (798) */
/* 846 */
        0x1d,    /* FC_SMFARRAY */
        0x0,     /* 0 */
/* 848 */NdrFcShort( 0x8 ),   /* 8 */
/* 850 */0x1,      /* FC_BYTE */
        0x5b,    /* FC_END */
/* 852 */
        0x15,    /* FC_STRUCT */
        0x3,     /* 3 */
/* 854 */NdrFcShort( 0x10 ), /* 16 */
/* 856 */0x8,      /* FC_LONG */
        0x6,     /* FC_SHORT */
/* 858 */0x6,      /* FC_SHORT */
        0x4c,    /* FC_EMBEDDED_COMPLEX */
/* 860 */0x0,      /* 0 */
        NdrFcShort( 0xfff1 ),   /* Offset= -15 (846) */
        0x5b,    /* FC_END */
/* 864 */
        0x1a,    /* FC_BOGUS_STRUCT */
        0x3,     /* 3 */
/* 866 */NdrFcShort( 0x18 ), /* 24 */
/* 868 */NdrFcShort( 0x0 ),   /* 0 */
/* 870 */NdrFcShort( 0xa ),   /* Offset= 10 (880) */
/* 872 */0x8,      /* FC_LONG */
        0x36,    /* FC_POINTER */
/* 874 */0x4c,     /* FC_EMBEDDED_COMPLEX */
        0x0,     /* 0 */
/* 876 */NdrFcShort( 0xffe8 ),  /* Offset= -24 (852) */
/* 878 */0x5c,     /* FC_PAD */
        0x5b,    /* FC_END */
/* 880 */
        0x11, 0x0,   /* FC_RP */
```

```
/* 882 */NdrFcShort( 0xfd4a ), /* Offset= -694 (188) */
/* 884 */
        0x1b,    /* FC_CARRAY */
        0x0,     /* 0 */
/* 886 */NdrFcShort( 0x1 ),   /* 1 */
/* 888 */0x19,    /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 890 */NdrFcShort( 0x0 ),   /* 0 */
/* 892 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 894 */0x1,     /* FC_BYTE */
        0x5b,    /* FC_END */
/* 896 */
        0x16,    /* FC_PSTRUCT */
        0x3,     /* 3 */
/* 898 */NdrFcShort( 0x8 ),   /* 8 */
/* 900 */
        0x4b,    /* FC_PP */
        0x5c,    /* FC_PAD */
/* 902 */
        0x46,    /* FC_NO_REPEAT */
        0x5c,    /* FC_PAD */
/* 904 */NdrFcShort( 0x4 ),   /* 4 */
/* 906 */NdrFcShort( 0x4 ),   /* 4 */
/* 908 */0x13, 0x0,   /* FC_OP */
/* 910 */NdrFcShort( 0xffe6 ),  /* Offset= -26 (884) */
/* 912 */
        0x5b,    /* FC_END */

        0x8,     /* FC_LONG */
/* 914 */0x8,     /* FC_LONG */
        0x5b,    /* FC_END */
/* 916 */
        0x1b,    /* FC_CARRAY */
        0x1,     /* 1 */
/* 918 */NdrFcShort( 0x2 ),   /* 2 */
/* 920 */0x19,    /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 922 */NdrFcShort( 0x0 ),   /* 0 */
/* 924 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 926 */0x6,     /* FC_SHORT */
        0x5b,    /* FC_END */
/* 928 */
        0x16,    /* FC_PSTRUCT */
        0x3,     /* 3 */
/* 930 */NdrFcShort( 0x8 ),   /* 8 */
/* 932 */
        0x4b,    /* FC_PP */
        0x5c,    /* FC_PAD */
/* 934 */
        0x46,    /* FC_NO_REPEAT */
        0x5c,    /* FC_PAD */
/* 936 */NdrFcShort( 0x4 ),   /* 4 */
/* 938 */NdrFcShort( 0x4 ),   /* 4 */
/* 940 */0x13, 0x0,   /* FC_OP */
/* 942 */NdrFcShort( 0xffe6 ),  /* Offset= -26 (916) */
/* 944 */
        0x5b,    /* FC_END */

        0x8,     /* FC_LONG */
/* 946 */0x8,     /* FC_LONG */
        0x5b,    /* FC_END */
/* 948 */
        0x1b,    /* FC_CARRAY */
        0x3,     /* 3 */
/* 950 */NdrFcShort( 0x4 ),   /* 4 */
/* 952 */0x19,    /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 954 */NdrFcShort( 0x0 ),   /* 0 */
/* 956 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 958 */0x8,     /* FC_LONG */
        0x5b,    /* FC_END */
/* 960 */
        0x16,    /* FC_PSTRUCT */
        0x3,     /* 3 */
/* 962 */NdrFcShort( 0x8 ),   /* 8 */
/* 964 */
        0x4b,    /* FC_PP */
        0x5c,    /* FC_PAD */
/* 966 */
        0x46,    /* FC_NO_REPEAT */
        0x5c,    /* FC_PAD */
/* 968 */NdrFcShort( 0x4 ),   /* 4 */
/* 970 */NdrFcShort( 0x4 ),   /* 4 */
/* 972 */0x13, 0x0,   /* FC_OP */
/* 974 */NdrFcShort( 0xffe6 ),  /* Offset= -26 (948) */
/* 976 */
        0x5b,    /* FC_END */

        0x8,     /* FC_LONG */
/* 978 */0x8,     /* FC_LONG */
        0x5b,    /* FC_END */
/* 980 */
        0x1b,    /* FC_CARRAY */
        0x7,     /* 7 */
/* 982 */NdrFcShort( 0x8 ),   /* 8 */
/* 984 */0x19,    /* Corr desc:  field pointer, FC_ULONG */
        0x0,     /*  */
/* 986 */NdrFcShort( 0x0 ),   /* 0 */
/* 988 */NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 990 */0xb,     /* FC_HYPER */
        0x5b,    /* FC_END */
/* 992 */
        0x16,    /* FC_PSTRUCT */
        0x3,     /* 3 */
/* 994 */NdrFcShort( 0x8 ),   /* 8 */
/* 996 */
        0x4b,    /* FC_PP */
        0x5c,    /* FC_PAD */
/* 998 */
        0x46,    /* FC_NO_REPEAT */
        0x5c,    /* FC_PAD */
/* 1000 */ NdrFcShort( 0x4 ),   /* 4 */
/* 1002 */ NdrFcShort( 0x4 ),   /* 4 */
/* 1004 */ 0x13, 0x0,   /* FC_OP */
/* 1006 */ NdrFcShort( 0xffe6 ),  /* Offset= -26 (980) */
/* 1008 */
        0x5b,    /* FC_END */

        0x8,     /* FC_LONG */
/* 1010 */ 0x8,     /* FC_LONG */
        0x5b,    /* FC_END */
/* 1012 */
        0x15,    /* FC_STRUCT */
        0x3,     /* 3 */
/* 1014 */ NdrFcShort( 0x8 ),   /* 8 */
/* 1016 */ 0x8,     /* FC_LONG */
        0x8,     /* FC_LONG */
/* 1018 */ 0x5c,    /* FC_PAD */
        0x5b,    /* FC_END */
/* 1020 */
        0x1b,    /* FC_CARRAY */
        0x3,     /* 3 */
/* 1022 */ NdrFcShort( 0x8 ),   /* 8 */
/* 1024 */ 0x7,     /* Corr desc: FC_USHORT */
        0x0,     /*  */
/* 1026 */ NdrFcShort( 0xffd8 ),  /* -40 */
/* 1028 */ NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 1030 */ 0x4c,    /* FC_EMBEDDED_COMPLEX */
        0x0,     /* 0 */
/* 1032 */ NdrFcShort( 0xffec ),  /* Offset= -20 (1012) */
/* 1034 */ 0x5c,    /* FC_PAD */
        0x5b,    /* FC_END */
/* 1036 */
        0x1a,    /* FC_BOGUS_STRUCT */
        0x3,     /* 3 */
/* 1038 */ NdrFcShort( 0x28 ), /* 40 */
/* 1040 */ NdrFcShort( 0xffec ),  /* Offset= -20 (1020) */
/* 1042 */ NdrFcShort( 0x0 ),   /* Offset= 0 (1042) */
/* 1044 */ 0x6,     /* FC_SHORT */
        0x6,     /* FC_SHORT */
/* 1046 */ 0x8,     /* FC_LONG */
        0x8,     /* FC_LONG */
/* 1048 */ 0x4c,    /* FC_EMBEDDED_COMPLEX */
        0x0,     /* 0 */
/* 1050 */ NdrFcShort( 0xfc18 ), /* Offset= -1000 (50) */
/* 1052 */ 0x5c,    /* FC_PAD */
```

```
        0x5b,    /* FC_END */
/* 1054 */ 0xb4,    /* FC_USER_MARSHAL */
        0x83,    /* 131 */
/* 1056 */ NdrFcShort( 0x1 ),   /* 1 */
/* 1058 */ NdrFcShort( 0x4 ),   /* 4 */
/* 1060 */ NdrFcShort( 0x0 ),   /* 0 */
/* 1062 */ NdrFcShort( 0xfc04 ), /* Offset= -1020 (42) */
/* 1064 */
        0x11, 0x8,   /* FC_RP [simple_pointer] */
/* 1066 */ 0x8,     /* FC_LONG */
        0x5c,    /* FC_PAD */
/* 1068 */
        0x11, 0x14, /* FC_RP [alloced_on_stack]
[pointer_deref] */
/* 1070 */ NdrFcShort( 0x2 ),   /* Offset= 2 (1072) */
/* 1072 */
        0x13, 0x0,   /* FC_OP */
/* 1074 */ NdrFcShort( 0x2 ),   /* Offset= 2 (1076) */
/* 1076 */
        0x1b,    /* FC_CARRAY */
        0x0,     /* 0 */
/* 1078 */ NdrFcShort( 0x1 ),   /* 1 */
/* 1080 */ 0x28,    /* Corr desc:  parameter, FC_LONG */
        0x54,    /* FC_DEREFERENCE */
/* 1082 */ NdrFcShort( 0x4 ),   /* x86 Stack size/offset = 4 */
/* 1084 */ NdrFcShort( 0x1 ),   /* Corr flags:  early, */
/* 1086 */ 0x2,     /* FC_CHAR */
        0x5b,    /* FC_END */

        0x0
      }
    };

static const USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[ WIRE_MARSHAL_TABLE_SIZE ] =
      {

          {
          BSTR_UserSize
          ,BSTR_UserMarshal
          ,BSTR_UserUnmarshal
          ,BSTR_UserFree
          },
          {
          LPSAFEARRAY_UserSize
          ,LPSAFEARRAY_UserMarshal
          ,LPSAFEARRAY_UserUnmarshal
          ,LPSAFEARRAY_UserFree
          }

      };
```

```
/* Object interface: IUnknown, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0
x00,0x00,0x00,0x46}} */


/* Object interface: IDispatch, ver. 0.0,

GUID={0x00020400,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0
x00,0x00,0x00,0x46}} */


/* Object interface: IComponentRegistrar, ver. 0.0,

GUID={0xa817e7a2,0x43fa,0x11d0,{0x9e,0x44,0x00,0xaa,0x
00,0xb6,0x77,0x0a}} */

#pragma code_seg(".orpc")
static const unsigned short
IComponentRegistrar_FormatStringOffsetTable[] =
    {
    (unsigned short) -1,
    (unsigned short) -1,
    (unsigned short) -1,
    (unsigned short) -1,
    0,
    36,
    66,
    96,
    138,
    174
    };

static const MIDL_STUBLESS_PROXY_INFO
IComponentRegistrar_ProxyInfo =
    {
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &IComponentRegistrar_FormatStringOffsetTable[-3],
    0,
    0,
    0
    };


static const MIDL_SERVER_INFO
IComponentRegistrar_ServerInfo =
    {
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &IComponentRegistrar_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0};
CINTERFACE_PROXY_VTABLE(13)
_IComponentRegistrarProxyVtbl =
{
    &IComponentRegistrar_ProxyInfo,
    &IID_IComponentRegistrar,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    0 /* (void *) (INT_PTR) -1 /* IDispatch::GetTypeInfoCount
*/ ,
    0 /* (void *) (INT_PTR) -1 /* IDispatch::GetTypeInfo */ ,
    0 /* (void *) (INT_PTR) -1 /* IDispatch::GetIDsOfNames */ ,
    0 /* IDispatch_Invoke_Proxy */ ,
    (void *) (INT_PTR) -1 /* IComponentRegistrar::Attach */ ,
    (void *) (INT_PTR) -1 /* IComponentRegistrar::RegisterAll
*/ ,
    (void *) (INT_PTR) -1 /*
IComponentRegistrar::UnregisterAll */ ,
    (void *) (INT_PTR) -1 /*
IComponentRegistrar::GetComponents */ ,
    (void *) (INT_PTR) -1 /*
IComponentRegistrar::RegisterComponent */ ,
    (void *) (INT_PTR) -1 /*
IComponentRegistrar::UnregisterComponent */
};


static const PRPC_STUB_FUNCTION
IComponentRegistrar_table[] =
{
    STUB_FORWARDING_FUNCTION,
    STUB_FORWARDING_FUNCTION,
    STUB_FORWARDING_FUNCTION,
    STUB_FORWARDING_FUNCTION,
    NdrStubCall2,
    NdrStubCall2,
    NdrStubCall2,
    NdrStubCall2,
    NdrStubCall2,
    NdrStubCall2
};

CInterfaceStubVtbl _IComponentRegistrarStubVtbl =
{
    &IID_IComponentRegistrar,
    &IComponentRegistrar_ServerInfo,
```

```
    13,
    &IComponentRegistrar_table[-3],
    CStdStubBuffer_DELEGATING_METHODS
};


/* Object interface: Itpcc_com, ver. 0.0,

GUID={0x5B4FA473,0x2E68,0x4D79,{0xA6,0x26,0xF3,0x8B,
0x30,0xB8,0x19,0x6E}} */

#pragma code_seg(".orpc")
static const unsigned short
Itpcc_com_FormatStringOffsetTable[] =
    {
    210,
    252,
    294,
    336,
    378,
    36
    };

static const MIDL_STUBLESS_PROXY_INFO
Itpcc_com_ProxyInfo =
    {
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &Itpcc_com_FormatStringOffsetTable[-3],
    0,
    0,
    0
    };


static const MIDL_SERVER_INFO Itpcc_com_ServerInfo =
    {
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &Itpcc_com_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0};
CINTERFACE_PROXY_VTABLE(9) _Itpcc_comProxyVtbl =
{
    &Itpcc_com_ProxyInfo,
    &IID_Itpcc_com,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    (void *) (INT_PTR) -1 /* Itpcc_com::doStockLevel */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com::doNewOrder */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com::doPayment */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com::doOrderStatus */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com::doDBInfo */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com::doSetComplete */
};

const CInterfaceStubVtbl _Itpcc_comStubVtbl =
{
    &IID_Itpcc_com,
    &Itpcc_com_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 * _tpccCom_ProxyVtblList[] =
{
    ( CInterfaceProxyVtbl *) &_Itpcc_comProxyVtbl,
    ( CInterfaceProxyVtbl *) &_IComponentRegistrarProxyVtbl,
    0
};

const CInterfaceStubVtbl * _tpccCom_StubVtblList[] =
{
    ( CInterfaceStubVtbl *) &_Itpcc_comStubVtbl,
    ( CInterfaceStubVtbl *) &_IComponentRegistrarStubVtbl,
    0
};

PCInterfaceName const _tpccCom_InterfaceNamesList[] =
{
    "Itpcc_com",
    "IComponentRegistrar",
    0
};

const IID *  _tpccCom_BaseIIDList[] =
{
    0,
    &IID_IDispatch,
    0
};


#define _tpccCom_CHECK_IID(n)
   IID_GENERIC_CHECK_IID( _tpccCom, pIID, n)

int __stdcall _tpccCom_IID_Lookup( const IID * pIID, int *
pIndex )
{
    IID_BS_LOOKUP_SETUP

    IID_BS_LOOKUP_INITIAL_TEST( _tpccCom, 2, 1 )
    IID_BS_LOOKUP_RETURN_RESULT( _tpccCom, 2,
*pIndex )

}

const ExtendedProxyFileInfo tpccCom_ProxyFileInfo =
{
    (PCInterfaceProxyVtblList *) & _tpccCom_ProxyVtblList,
    (PCInterfaceStubVtblList *) & _tpccCom_StubVtblList,
    (const PCInterfaceName * ) &
_tpccCom_InterfaceNamesList,
    (const IID ** ) & _tpccCom_BaseIIDList,
    & _tpccCom_IID_Lookup,
    2,
    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)*/
```

tpccCom.rc

```
// Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "winres.h"

//////////////////////////////////////////////////////////////////////////////
#undef APSTUDIO_READONLY_SYMBOLS

//////////////////////////////////////////////////////////////////////////////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) ||
defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifdef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE
BEGIN
   "resource.h\0"
END

2 TEXTINCLUDE
BEGIN
   "#include ""winres.h""\r\n"
   "\0"
END

3 TEXTINCLUDE
BEGIN
   "1 TYPELIB ""tpccCom.tlb""\r\n"
   "\0"
END
#endif    // APSTUDIO_INVOKED


//////////////////////////////////////////////////////////////////////////////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 1,0,0,1
 PRODUCTVERSION 1,0,0,1
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x4L
 FILETYPE 0x2L
 FILESUBTYPE 0x0L
BEGIN
   BLOCK "StringFileInfo"
   BEGIN
      BLOCK "040904e4"
      BEGIN
         VALUE "CompanyName", "TODO: <Company
name>"
         VALUE "FileDescription", "TODO: <File description>"
         VALUE "FileVersion", "1.0.0.1"
         VALUE "LegalCopyright", "TODO: (c) <Company
name>.  All rights reserved."
         VALUE "InternalName", "tpccCom.dll"
         VALUE "OriginalFilename", "tpccCom.dll"
         VALUE "ProductName", "TODO: <Product name>"
         VALUE "ProductVersion", "1.0.0.1"
      END
   END
   BLOCK "VarFileInfo"
   BEGIN
      VALUE "Translation", 0x409, 1252
   END
END


//////////////////////////////////////////////////////////////////////////////
//
// REGISTRY
//

IDR_TPCCCOM            REGISTRY            "tpccCom.rgs"
IDR_TPCC_COM           REGISTRY
"tpcc_com.rgs"


//////////////////////////////////////////////////////////////////////////////
//
// String Table
//

STRINGTABLE
BEGIN
   IDS_PROJNAME           "tpccCom"
END

#endif    // English (U.S.) resources
//////////////////////////////////////////////////////////////////////////////



#ifndef APSTUDIO_INVOKED
//////////////////////////////////////////////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
1 TYPELIB "tpccCom.tlb"

//////////////////////////////////////////////////////////////////////////////
#endif    // not APSTUDIO_INVOKED
```

tpcc_com.rgs

```
HKCR
{
  tpccCom.tpcc_com.1 = s 'tpcc_com Class'
  {
    CLSID = s '{5F752BF2-F739-43D4-8492-
44C19581C0A1}'
  }
  tpccCom.tpcc_com = s 'tpcc_com Class'
  {
    CLSID = s '{5F752BF2-F739-43D4-8492-
44C19581C0A1}'
    CurVer = s 'tpccCom.tpcc_com.1'
  }
  NoRemove CLSID
  {
    ForceRemove {5F752BF2-F739-43D4-8492-
44C19581C0A1} = s 'tpcc_com Class'
    {
      ProgID = s 'tpccCom.tpcc_com.1'
      VersionIndependentProgID = s 'tpccCom.tpcc_com'
      InprocServer32 = s '%MODULE%'
      {
        val ThreadingModel = s 'Both'
```

```
        }
        val AppID = s '%APPID%'
        'TypeLib' = s '{91F1B8B0-89E9-457B-A228-
3E2D6CE3E752}'
      }
   }
}
```

tpccCom.rgs

```
HKCR
{
   NoRemove AppID
   {
      '%APPID%' = s 'tpccCom'
      'tpccCom.DLL'
      {
         val AppID = s '%APPID%'
      }
   }
   NoRemove CLSID
   {
      ForceRemove {90EEDAFF-F8D3-4711-99A9-
8AC3C0FE5DB9} = s 'CompReg Class'
      {
         InprocServer32 = s '%MODULE%'
         {
            val ThreadingModel = s 'Apartment'
         }
         'TypeLib' = s '{91F1B8B0-89E9-457B-A228-
3E2D6CE3E752}'
      }
   }
}
```

11 Appendix - B: Tunable Parameters

11.1. Database Parameters

run_15_0.cfg

```
#################################################
#############################
#
#    Configuration File for the Sybase SQL Server
#
#    Please read the System Administration Guide (SAG)
#    before changing any of the values in this file.
#
#################################################
#############################


[Configuration Options]

[General Information]

[Backup/Recovery]
   recovery interval in minutes = 32767
   print recovery information = DEFAULT
   tape retention in days = DEFAULT
   max concurrently recovered db = DEFAULT
   number of checkpoint tasks = DEFAULT

[Cache Manager]
   number of oam trips = 5
   number of index trips = DEFAULT
   memory alignment boundary = DEFAULT
   global async prefetch limit = 0
   global cache partition number = 16
   extended cache size = DEFAULT

[Named Cache:c_customer]
   cache size = 700M
   cache status = mixed cache
   cache replacement policy = DEFAULT
   local cache partition number = DEFAULT

[4K I/O Buffer Pool]
   pool size = DEFAULT
   wash size = 16384 K
   local async prefetch limit = DEFAULT

[Named Cache:c_customer_index]
   cache size = 3600M
   cache status = mixed cache
   cache status = HK ignore cache
   cache replacement policy = DEFAULT
   local cache partition number = DEFAULT

[4K I/O Buffer Pool]
   pool size = DEFAULT
   wash size = 65536 K
   local async prefetch limit = DEFAULT

[Named Cache:c_index]
   cache size = 3000M
   cache status = mixed cache
   cache status = HK ignore cache
   cache replacement policy = DEFAULT
   local cache partition number = DEFAULT

[4K I/O Buffer Pool]
   pool size = DEFAULT
   wash size = 65536 K
   local async prefetch limit = DEFAULT

[Named Cache:c_log]
   cache size = 200M
   cache status = log only
   cache replacement policy = DEFAULT
   local cache partition number = 1

[8K I/O Buffer Pool]
   pool size = 190M
   wash size = 8192 K
   local async prefetch limit = DEFAULT

[Named Cache:c_no]
   cache size = 3000M
   cache status = mixed cache
   cache replacement policy = DEFAULT
   local cache partition number = DEFAULT

[4K I/O Buffer Pool]
   pool size = DEFAULT
   wash size = 65536 K
   local async prefetch limit = DEFAULT

[Named Cache:c_orders]
   cache size = 11600M
   cache status = mixed cache
   cache replacement policy = DEFAULT
   local cache partition number = 32

[4K I/O Buffer Pool]
   pool size = DEFAULT
   wash size = 65536 K
   local async prefetch limit = DEFAULT

[8K I/O Buffer Pool]
   pool size = 1500M
   wash size = 65536 K
   local async prefetch limit = DEFAULT

[Named Cache:c_stock]
   cache size = 96000M
   cache status = mixed cache
   cache replacement policy = DEFAULT
   local cache partition number = 64

[4K I/O Buffer Pool]
   pool size = DEFAULT
   wash size = 65536 K
   local async prefetch limit = DEFAULT

[Named Cache:c_tinyhot]
   cache size = 1200M
   cache status = mixed cache
   cache status = HK ignore cache
   cache replacement policy = relaxed LRU replacement
   local cache partition number = DEFAULT

[4K I/O Buffer Pool]
   pool size = DEFAULT
   wash size = 4096 K
   local async prefetch limit = DEFAULT

[Named Cache:default data cache]
   cache size = 200M
   cache status = default data cache
   cache status = HK ignore cache
   cache replacement policy = DEFAULT
   local cache partition number = DEFAULT

[4K I/O Buffer Pool]
   pool size = DEFAULT
   wash size = 4096 K
   local async prefetch limit = DEFAULT

[Meta-Data Caches]
   number of open databases = 6
   number of open objects = 200
```

```
   open object spinlock ratio = DEFAULT
   number of open indexes = 100
   open index hash spinlock ratio = DEFAULT
   open index spinlock ratio = DEFAULT
   partition groups = 20
   partition spinlock ratio = DEFAULT
   number of open partitions = 2048

[Disk I/O]
   disk i/o structures = 6000
   number of large i/o buffers = DEFAULT
   page utilization percent = DEFAULT
   number of devices = 160
   disable disk mirroring = DEFAULT
   allow sql server async i/o = DEFAULT

[Languages]
   disable character set conversions = DEFAULT

[Unicode]
   enable unicode normalization = 0
   enable surrogate processing = 0
   enable unicode conversions = 0
   size of unilib cache = DEFAULT

[Network Communication]
   default network packet size = 4096
   max network packet size = 4096
   remote server pre-read packets = DEFAULT
   number of remote connections = DEFAULT
   number of remote logins = DEFAULT
   number of remote sites = DEFAULT
   max number network listeners = DEFAULT
   tcp no delay = DEFAULT
   send doneinproc tokens = 0
   allow sendmsg = DEFAULT
   syb_sendmsg port number = DEFAULT
   allow remote access = DEFAULT

[O/S Resources]
   max async i/os per engine = 8000
   max async i/os per server = 16000

[Query Tuning]
   optimization goal = DEFAULT
   allow backward scans = DEFAULT
   abstract plan load = DEFAULT
   abstract plan dump = DEFAULT
   abstract plan replace = DEFAULT
   abstract plan cache = DEFAULT
   sampling percent = DEFAULT
   number of histogram steps = DEFAULT
   enable sort-merge join and JTC = DEFAULT
   number of worker processes = DEFAULT
   memory per worker process = DEFAULT
   max parallel degree = DEFAULT
   max scan parallel degree = DEFAULT
   max repartition degree = DEFAULT
   max resource granularity = DEFAULT
   enable metrics capture = DEFAULT
   optimization timeout limit = DEFAULT
   metrics lio max = DEFAULT
   metrics pio max = DEFAULT
   metrics elap max = DEFAULT
   metrics exec max = DEFAULT
   sproc optimize timeout limit = DEFAULT
   min pages for parallel scan = DEFAULT
   prod-consumer overlap factor = DEFAULT
   enable literal autoparam = DEFAULT
   max query parallel degree = DEFAULT
   cost of a logical io = DEFAULT
   cost of a physical io = DEFAULT
   cost of a cpu unit = DEFAULT
   auto query tuning = DEFAULT
   enable query tuning mem limit = DEFAULT
   query tuning plan executions = DEFAULT
   enable query tuning time limit = DEFAULT
   max buffers per lava operator = DEFAULT

[Physical Resources]

[Physical Memory]
   max memory = 65273856
   additional network memory = DEFAULT
   shared memory starting address = DEFAULT
   allocate max shared memory = 1
   dynamic allocation on demand = DEFAULT
   lock shared memory = 1
   heap memory per user = DEFAULT
   engine memory log size = DEFAULT
   compression memory size = DEFAULT

[Processors]
   max online engines = 8
   number of engines at startup = 8
   statement cache size = DEFAULT

[SQL Server Administration]
   procedure cache size = 296320
   default database size = DEFAULT
   identity burning set factor = DEFAULT
   allow nested triggers = DEFAULT
   allow updates to system tables = 1
   default fill factor percent = DEFAULT
   default exp_row_size percent = DEFAULT
   number of mailboxes = DEFAULT
   number of messages = DEFAULT
   number of alarms = DEFAULT
   number of pre-allocated extents = DEFAULT
   event buffers per engine = DEFAULT
   cpu accounting flush interval = DEFAULT
   i/o accounting flush interval = DEFAULT
   sql server clock tick length = DEFAULT
   runnable process search count = 100
   i/o polling process count = DEFAULT
   time slice = DEFAULT
   cpu grace time = DEFAULT
   number of sort buffers = DEFAULT
   size of auto identity column = DEFAULT
   identity grab size = DEFAULT
   housekeeper free write percent = 0
   enable housekeeper GC = 0
   sysstatistics flush interval = DEFAULT
   allow resource limits = DEFAULT
   number of aux scan descriptors = DEFAULT
   SQL Perfmon Integration = DEFAULT
   license information = DEFAULT
   text prefetch size = DEFAULT
   enable HA = DEFAULT
   i/o batch size = 15
   enable semantic partitioning = DEFAULT
   enable xml = DEFAULT
   enable webservices = DEFAULT
   enable job scheduler = DEFAULT
   job scheduler tasks = DEFAULT
   job scheduler interval = DEFAULT
   percent database for history = DEFAULT
   percent history free = DEFAULT
   percent database for output = DEFAULT
   percent output free = DEFAULT
   maximum job output = DEFAULT
   enable sql debugger = DEFAULT

[User Environment]
   number of user connections = 500
   stack size = DEFAULT
   stack guard size = DEFAULT
   permission cache entries = DEFAULT
   user log cache size = DEFAULT
   user log cache spinlock ratio = DEFAULT
   session tempdb log cache size = DEFAULT
   max native threads per engine = DEFAULT
   messaging memory = DEFAULT
   enable real time messaging = DEFAULT
   histogram tuning factor = DEFAULT
   rtm thread idle wait period = DEFAULT
```

```
[Lock Manager]
   number of locks = 15000
   deadlock checking period = 1000
   lock spinlock ratio = DEFAULT
   lock address spinlock ratio = DEFAULT
   lock table spinlock ratio = 1
   lock hashtable size = 16384
   lock scheme = DEFAULT
   lock wait period = DEFAULT
   read committed with lock = DEFAULT
   print deadlock information = DEFAULT
   deadlock retries = DEFAULT
   page lock promotion HWM = DEFAULT
   page lock promotion LWM = DEFAULT
   page lock promotion PCT = DEFAULT
   row lock promotion HWM = DEFAULT
   row lock promotion LWM = DEFAULT
   row lock promotion PCT = DEFAULT

[Security Related]
   systemwide password expiration = DEFAULT
   audit queue size = DEFAULT
   curread change w/ open cursors = DEFAULT
   allow procedure grouping = DEFAULT
   select on syscomments.text = DEFAULT
   auditing = DEFAULT
   current audit table = DEFAULT
   suspend audit when device full = DEFAULT
   enable row level access = DEFAULT
   check password for digit = DEFAULT
   minimum password length = DEFAULT
   maximum failed logins = DEFAULT
   enable ssl = DEFAULT
   unified login required = DEFAULT
   use security services = DEFAULT
   msg confidentiality reqd = DEFAULT
   msg integrity reqd = DEFAULT
   enable pam user auth = DEFAULT
   enable ldap user auth = DEFAULT
   enable encrypted columns = DEFAULT
   secure default login = DEFAULT
   enable logins during recovery = DEFAULT

[Extended Stored Procedure]
   esp unload dll = DEFAULT
   esp execution priority = DEFAULT
   esp execution stacksize = DEFAULT
   xp_cmdshell context = DEFAULT
   start mail session = DEFAULT
   start xp server during boot = DEFAULT

[Error Log]
   event logging = 0
   log audit logon success = DEFAULT
   log audit logon failure = DEFAULT
   event log computer name = DEFAULT

[Rep Agent Thread Administration]
   enable rep agent threads = DEFAULT

[Component Integration Services]
   enable cis = 0
   cis connect timeout = DEFAULT
   cis bulk insert batch size = DEFAULT
   max cis remote connections = DEFAULT
   cis idle connection timeout = DEFAULT
   cis packet size = DEFAULT
   cis cursor rows = DEFAULT
   enable snmp = DEFAULT
   enable file access = DEFAULT
   cis bulk insert array size = DEFAULT
   enable full-text search = DEFAULT
   cis rpc handling = DEFAULT

[Java Services]
   enable java = DEFAULT
   size of process object heap = DEFAULT
   size of shared class heap = DEFAULT
   size of global fixed heap = DEFAULT
   number of java sockets = DEFAULT

[DTM Administration]
   enable DTM = DEFAULT
   enable xact coordination = 0
   xact coordination interval = DEFAULT
   number of dtx participants = DEFAULT
   strict dtm enforcement = DEFAULT
   txn to pss ratio = DEFAULT
   dtm lock timeout period = DEFAULT
   dtm detach timeout period = DEFAULT

[Diagnostics]
   dump on conditions = DEFAULT
   maximum dump conditions = DEFAULT
   number of dump threads = DEFAULT
   number of ccbs = DEFAULT
   caps per ccb = DEFAULT
   average cap size = DEFAULT

[Monitoring]
   enable monitoring = DEFAULT
   sql text pipe active = DEFAULT
   sql text pipe max messages = DEFAULT
   plan text pipe active = DEFAULT
   plan text pipe max messages = DEFAULT
   statement pipe active = DEFAULT
   statement pipe max messages = DEFAULT
   errorlog pipe active = DEFAULT
   errorlog pipe max messages = DEFAULT
   deadlock pipe active = DEFAULT
   deadlock pipe max messages = DEFAULT
   wait event timing = DEFAULT
   process wait events = DEFAULT
   object lockwait timing = DEFAULT
   SQL batch capture = DEFAULT
   statement statistics active = DEFAULT
   per object statistics active = DEFAULT
   max SQL text monitored = DEFAULT
   performance monitoring option = DEFAULT
   enable stmt cache monitoring = DEFAULT
   identity reservation size = DEFAULT
   max online Q engines = DEFAULT
   number of Q engines at startup = DEFAULT
   net password encryption reqd = DEFAULT
   restricted decrypt permission = DEFAULT
   enable merge join = DEFAULT
```

tpcc_cache_bind.sh

```
#!/bin/sh -f
isql -Usa -P$PASSWORD << EOF
dbcc tune("max_eng_freelocks", 30)
go
dbcc tune("washbatchsize", 20)
go
dbcc tune("io_getevents_timeout", 0)
go
use master
go
sp_configure "i/o batch size", 15
go
sp_configure "disable disk mirroring", 1
go
sp_configure "esp unload dll", 0
go
sp_configure "enable java", 0
go
sp_configure "enable DTM", 0
go
sp_configure "enable monitoring", 0
go
sp_dboption 'tpcc', 'delayed commit', false
go
sp_dboption tpcc, "single user", true
go
```

```
use tpcc
go
checkpoint
go

sp_bindcache "c_customer", "tpcc", "customer"
go
sp_bindcache "c_stock", "tpcc", "stock"
go
sp_bindcache "c_tinyhot", "tpcc", "sysindexes", "sysindexes"
go
sp_bindcache "c_tinyhot", "tpcc", "warehouse", "w_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "district", "d_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "item", "i_clu"
go
sp_bindcache "c_tinyhot", "tpcc", "sysindexes"
go
sp_bindcache "c_tinyhot", "tpcc", "warehouse"
go
sp_bindcache "c_tinyhot", "tpcc", "district"
go
sp_bindcache "c_tinyhot", "tpcc", "item"
go
sp_bindcache "c_index", "tpcc", "customer", "c_clu"
go
sp_bindcache "c_customer_index", "tpcc", "customer",
"c_non1"
go
sp_bindcache "c_index", "tpcc", "new_order", "no_clu"
go
sp_bindcache "c_index", "tpcc", "orders", "o_clu"
go
sp_bindcache "c_index", "tpcc", "order_line", "ol_clu"
go
sp_bindcache "c_log", "tpcc", "syslogs"
go
sp_bindcache "c_no", "tpcc", "new_order"
go
sp_bindcache "c_orders", "tpcc", "order_line"
go
sp_bindcache "c_orders", "tpcc", "orders"
go
sp_bindcache "c_history", "tpcc", "history"
go

use master
go
sp_dboption tpcc, "single user", false
go
```

```
use tpcc
go
checkpoint
go

EOF
```

tpcc_desbind.sh

```
#
# script to bind the hot objects and procedures to their own
des
#
isql -Usa -P$PASSWORD << EOF
declare @dbid  int
select @dbid = db_id("tpcc")
dbcc tune(des_bind, @dbid, warehouse)
dbcc tune(des_bind, @dbid, district)
dbcc tune(des_bind, @dbid, item)
dbcc tune(des_bind, @dbid, stock)
dbcc tune(des_bind, @dbid, order_line)
dbcc tune(des_bind, @dbid, orders)
dbcc tune(des_bind, @dbid, new_order)
dbcc tune(des_bind, @dbid, customer)
dbcc tune(des_bind, @dbid, history)
dbcc tune(des_bind, @dbid, neworder_local)
dbcc tune(des_bind, @dbid, neworder_remote)
dbcc tune(des_bind, @dbid, payment_byid)
dbcc tune(des_bind, @dbid, payment_byname)
dbcc tune(des_bind, @dbid, order_status_byid)
dbcc tune(des_bind, @dbid, order_status_byname)
dbcc tune(des_bind, @dbid, delivery)
dbcc tune(des_bind, @dbid, stock_level)
go
EOF
```

setup_enginegroup_4groups.sql

```
sp_addengine 0, TPCC_0
go
sp_addengine 1, TPCC_0
go
sp_addexeclass "TPCC_0", "MEDIUM", 0, "TPCC_0"
go
sp_bindexeclass 'TPCC_0', 'ap', NULL, 'TPCC_0'
go

sp_addengine 2, TPCC_1
go
sp_addengine 3, TPCC_1
go
sp_addexeclass "TPCC_1", "MEDIUM", 0, "TPCC_1"
go
sp_bindexeclass 'TPCC_1', 'ap', NULL, 'TPCC_1'
go

sp_addengine 4, TPCC_2
go
sp_addengine 5, TPCC_2
go
sp_addexeclass "TPCC_2", "MEDIUM", 0, "TPCC_2"
go
sp_bindexeclass 'TPCC_2', 'ap', NULL, 'TPCC_2'
go

sp_addengine 6, TPCC_3
go
sp_addengine 7, TPCC_3
go
sp_addexeclass "TPCC_3", "MEDIUM", 0, "TPCC_3"
go
sp_bindexeclass 'TPCC_3', 'ap', NULL, 'TPCC_3'
go

sp_showexeclass
go

sp_helpgroup
go

sp_showpsexe
go
```

go_bind_sybase_4cpus.sh

```
ps -ef | grep "/bin/dataserver" | grep -v "grep" | grep -E
"ONLINE|master" > tmp1
gawk '{ print $1 " " $2 }' tmp1 > tmp2
gawk '{ if (NR == 1) sub(/sybase/,"taskset -p 03"); print }'
tmp2 > tmp3
gawk '{ if (NR == 2) sub(/sybase/,"taskset -p 03"); print }'
tmp2 >> tmp3
gawk '{ if (NR == 3) sub(/sybase/,"taskset -p 0c"); print }'
tmp2 >> tmp3
gawk '{ if (NR == 4) sub(/sybase/,"taskset -p 0c"); print }'
tmp2 >> tmp3
gawk '{ if (NR == 5) sub(/sybase/,"taskset -p 30"); print }'
tmp2 >> tmp3
gawk '{ if (NR == 6) sub(/sybase/,"taskset -p 30"); print }'
tmp2 >> tmp3
gawk '{ if (NR == 7) sub(/sybase/,"taskset -p c0"); print }'
tmp2 >> tmp3
```

```
gawk '{ if (NR == 8) sub(/sybase/,"taskset -p c0"); print }'
tmp2 >> tmp3
grep taskset tmp3 > bind_sybase.sh
chmod +x bind_sybase.sh
./bind_sybase.sh
rm tmp1 tmp2 tmp3
```

11.2. Transaction Monitor Parameters

inetInfo_registry.reg

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters]
"ListenBackLog"=dword:00000040
"DispatchEntries"=hex(7):00,00,13,04,9d,3d,83,7c,40,9d,88,7c,cc,8b,13,04,00,00,\
  00,00
"PoolThreadLimit"=dword:00000320
"MaxConcurrency"=dword:00000320
"MaxPoolThreads"=dword:000000c8
```

tcpip_parameters_registery.reg

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters]
"NV Hostname"="client1"
"DataBasePath"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,6f,\
  00,74,00,25,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,\
  64,00,72,00,69,00,76,00,65,00,72,00,73,00,5c,00,65,00,74,00,63,00,00,00
"NameServer"=""
"ForwardBroadcasts"=dword:00000000
"IPEnableRouter"=dword:00000000
"Domain"=""
"Hostname"="client1"
"SearchList"=""
"UseDomainNameDevolution"=dword:00000001
"EnableICMPRedirect"=dword:00000001
"DeadGWDetectDefault"=dword:00000001
"DontAddDefaultGatewayDefault"=dword:00000000
"EnableSecurityFilters"=dword:00000000
"EnableTCPA"=dword:00000001
"EnableRSS"=dword:00000001
"EnableTCPChimney"=dword:00000001
"MaxUserPort"=dword:00009c40

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapters]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapters\NdisWanIp]
"LLInterface"="WANARP"
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
  6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
  00,61,00,63,00,65,00,73,00,5c,00,7b,00,44,00,36,00,33,00,44,00,39,00,41,00,\
  33,00,35,00,2d,00,42,00,33,00,33,00,41,00,2d,00,34,00,32,00,41,00,32,00,2d,\
  00,41,00,44,00,42,00,41,00,2d,00,38,00,42,00,43,00,35,00,37,00,45,00,32,00,\
  31,00,38,00,31,00,31,00,31,00,7d,00,00,00,54,00,63,00,70,00,69,00,70,00,5c,\
  00,50,00,61,00,72,00,61,00,6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,\
  6e,00,74,00,65,00,72,00,66,00,61,00,63,00,65,00,73,00,5c,00,7b,00,45,00,33,\
  00,42,00,37,00,46,00,44,00,44,00,33,00,2d,00,31,00,44,00,45,00,32,00,2d,00,\
  34,00,45,00,43,00,45,00,2d,00,41,00,37,00,39,00,37,00,2d,00,35,00,39,00,35,\
  00,36,00,41,00,42,00,39,00,46,00,31,00,36,00,35,00,45,00,7d,00,00,00,00,00
"NumInterfaces"=dword:00000002
"IpInterfaces"=hex:35,9a,3d,d6,3a,b3,a2,42,ad,ba,8b,c5,7e,21,81,11,d3,fd,b7,e3,\
  e2,1d,ce,4e,a7,97,59,56,ab,9f,16,5e

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapters\{7F222411-5093-4B93-A63B-2996E3B2B1AC}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
  6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
  00,61,00,63,00,65,00,73,00,5c,00,7b,00,37,00,46,00,32,00,32,00,32,00,34,00,\
  31,00,31,00,2d,00,35,00,30,00,39,00,33,00,2d,00,34,00,42,00,39,00,33,00,2d,\
  00,41,00,36,00,33,00,42,00,2d,00,32,00,39,00,39,00,36,00,45,00,33,00,42,00,\
  32,00,42,00,31,00,41,00,43,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Adapters\{9862DFFF-C200-43E7-A141-BC26900FD0FC}]
"LLInterface"=""
"IpConfig"=hex(7):54,00,63,00,70,00,69,00,70,00,5c,00,50,00,61,00,72,00,61,00,\
  6d,00,65,00,74,00,65,00,72,00,73,00,5c,00,49,00,6e,00,74,00,65,00,72,00,66,\
  00,61,00,63,00,65,00,73,00,5c,00,7b,00,39,00,38,00,36,00,32,00,44,00,46,00,\
  46,00,46,00,2d,00,43,00,32,00,30,00,30,00,2d,00,34,00,33,00,45,00,37,00,2d,\
  00,41,00,31,00,34,00,31,00,2d,00,42,00,43,00,32,00,36,00,39,00,30,00,30,00,\
  46,00,44,00,30,00,46,00,43,00,7d,00,00,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\DNSRegisteredAdapters]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{7F222411-5093-4B93-A63B-2996E3B2B1AC}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):31,00,39,00,32,00,2e,00,31,00,36,00,38,00,2e,00,31,00,31,00,\
  2e,00,32,00,00,00,00,00
"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,\
  00,35,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00
,\
  33,00,00,00,00,00
"DhcpClassIdBin"=hex:
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000e10
"LeaseObtainedTime"=dword:46a8f3ab
"T1"=dword:46a8fab3
"T2"=dword:46a8fff9
"LeaseTerminatesTime"=dword:46a901bb
"IPAutoconfigurationAddress"="0.0.0.0"
"IPAutoconfigurationMask"="255.255.0.0"
"IPAutoconfigurationSeed"=dword:00000000
"AddressType"=dword:00000000
"MTU"=dword:00002328

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{9862DFFF-C200-43E7-A141-BC26900FD0FC}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,31,00,2e,00,31,00,2e,00,32,00,00,00,00,0
0
"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,
\
  00,35,00,2e,00,30,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"RegistrationEnabled"=dword:00000001
"RegisterAdapterName"=dword:00000000
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00,30,00
,\
  32,00,00,00,00,00
"DhcpClassIdBin"=hex:
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000e10
"LeaseObtainedTime"=dword:46a7b63e
"T1"=dword:46a7bd46
"T2"=dword:46a7c28c
"LeaseTerminatesTime"=dword:46a7c44e
"IPAutoconfigurationAddress"="0.0.0.0"
"IPAutoconfigurationMask"="255.255.0.0"
"IPAutoconfigurationSeed"=dword:00000000
"AddressType"=dword:00000000
"TcpWindowSize"=dword:00040000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{D63D9A35-B33A-42A2-ADBA-8BC57E218111}]
"UseZeroBroadcast"=dword:00000000
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00,00
"SubnetMask"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"EnableDeadGWDetect"=dword:00000001
"DontAddDefaultGateway"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{E3B7FDD3-1DE2-4ECE-A797-5956AB9F165E}]
"UseZeroBroadcast"=dword:00000000
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00,00
"SubnetMask"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"EnableDeadGWDetect"=dword:00000001
"DontAddDefaultGateway"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\PersistentRoutes]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Winsock]
"UseDelayedAcceptance"=dword:00000000
"HelperDllName"=hex(2):25,00,53,00,79,00,73,00,74,00,65,00,6d,00,52,00,6f,00,\
  6f,00,74,00,25,00,5c,00,53,00,79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,\
```

```
  00,77,00,73,00,68,00,74,00,63,00,70,00,69,00,70,00,2e,00,64,00,6c,00,6c,00,\
  00,00
"MaxSockAddrLength"=dword:00000010
"MinSockAddrLength"=dword:00000010
"Mapping"=hex:0b,00,00,00,03,00,00,00,02,00,00,00,01,00,00,00,06,00,00,00,02,
\
  00,00,00,01,00,00,00,00,00,00,00,02,00,00,00,00,00,00,00,06,00,00,00,00,00,\
  00,00,00,00,00,00,06,00,00,00,00,00,00,00,01,00,00,00,06,00,00,00,02,00,00,\
  00,02,00,00,00,11,00,00,00,02,00,00,00,02,00,00,00,00,00,00,00,02,00,00,00,\
  00,00,00,00,11,00,00,00,00,00,00,00,00,00,00,11,00,00,00,00,00,00,00,02,\
  00,00,00,11,00,00,00,02,00,00,00,03,00,00,00,00,00,00
```

tpccCom.tpcc_com_settingss.txt.txt

```
Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SOFTWARE\TPCC]
"dlvyLogPath"="c:\\InetPub\\wwwroot\\tpcc\\"
"dlvyQueueLen"=dword:00009c40
"dlvyThreads"=dword:00000007
"nullDB"=dword:00000000
"htmlTrace"=dword:00000000
"dbName"="tpcc"
"errorLogFile"="c:\\inetpub\\wwwroot\\tpcc\\isapi_err.log"
"htmlTraceLogFile"="c:\\inetpub\\wwwroot\\tpcc\\isapi.log"
"numUsers"=dword:00007530
"dbType"="SYBASE"
"dbUserName"="sa"
"dbPassword"=""
"isapi_trace"=dword:00000000
"numServers"=dword:00000001
"numWarehouse"=dword:000009c4
"numPools"=dword:00000001
"dbInterfacePath"="C:\\inetpub\\wwwroot\\tpcc\\tpccSybaseglue.dll"
"engineApp"="TPCC_0"
```

tpcc_software_registry.reg client1

```
Transactions not supported
Enable Object pooling
Minimum pool size 43
Maximum pool size 43
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required
```

tpcc_software_registry.reg client2

```
Transactions not supported
Enable Object pooling
Minimum pool size 43
Maximum pool size 43
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required
```

tpcc_software_registry.reg client3

```
Transactions not supported
Enable Object pooling
Minimum pool size 43
Maximum pool size 43
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required
```

tpcc_software_registry.reg client4

```
Transactions not supported
Enable Object pooling
Minimum pool size 43
Maximum pool size 43
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required
```

tpcc_software_registry.reg client5

```
Transactions not supported
Enable Object pooling
Minimum pool size 43
Maximum pool size 43
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required
```

tpcc_software_registry.reg client6

```
Transactions not supported
Enable Object pooling
Minimum pool size 43
Maximum pool size 43
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required
```

tpcc_software_registry.reg client7

```
Transactions not supported
Enable Object pooling
Minimum pool size 43
Maximum pool size 43
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required
```

tpcc_software_registry.reg client8

```
Transactions not supported
Enable Object pooling
Minimum pool size 43
Maximum pool size 43
Creation timeout 1,800,000,000
Enable Just in time activation
Concurrency Required
```

11.3. OS Configuration and Parameters

do_start_sybase

```
sleep 20

mount /dev/sdb1 /disk2
mount /dev/sdz1 /sybdmp1
mount /dev/sdaa1 /sybdmp2

ifconfig eth0 192.168.11.1 mask 255.255.255.0
ifconfig eth0 mtu 9000

ifconfig eth1 192.167.12.201 mask 255.255.255.0
ifconfig eth1 mtu 9000

chmod 777 /dev/sd[c-z]*

echo 503 > /proc/sys/vm/hugetlb_shm_group

echo 8000 > /proc/sys/vm/nr_hugepages

/root/setparams.sh > /root/setparams.log 2>&1
```

Disksetup.sh

```
#!/bin/ksh

disks="
/dev/sdc6 /dev/sdd6 /dev/sde6 /dev/sdf6 /dev/sdg6 /dev/sdh6
/dev/sdi6 /dev/sdj6 /dev/sdk6 /dev/sdl6 /dev/sdm6 /dev/sdn6
/dev/sdo6 /dev/sdp6 /dev/sdq6 /dev/sdr6 /dev/sds6 /dev/sdt6
/dev/sdu6 /dev/sdv6"

if true
then
for d in $disks
do
pvcreate --force $d
done
fi

vgcreate -s 4M -p 20 olvg $disks
for i in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20
do
lvcreate -i 20 -I 64k -L 31200M -n ol${i} olvg $disks
done

lvcreate -i 20 -I 64k -L 1420M -n ol21 olvg $disks
```

Setparams.sh

```
#!/bin/sh -x

vgscan
sleep 60

vgchange -a y olvg
```

```
sleep 10

awk '{
printf "raw %s %s\n", $2, $3
}' /root/diskmap | sh -x
sleep 2
awk '{
printf "ln -sf %s %s\n", $2, $1
}' /root/diskmap | sh -x
echo /dev/raw/* |xargs chown sybase
```

Diskmap

```
/tpcc/dat/rord1 /dev/raw/raw1 /dev/sdc1
/tpcc/dat/rord2 /dev/raw/raw2 /dev/sdd1
/tpcc/dat/rord3 /dev/raw/raw3 /dev/sde1
/tpcc/dat/rord4 /dev/raw/raw4 /dev/sdf1
/tpcc/dat/rord5 /dev/raw/raw5 /dev/sdg1
/tpcc/dat/rord6 /dev/raw/raw6 /dev/sdh1
/tpcc/dat/rord7 /dev/raw/raw7 /dev/sdi1
/tpcc/dat/rord8 /dev/raw/raw8 /dev/sdj1
/tpcc/dat/rord9 /dev/raw/raw9 /dev/sdk1
/tpcc/dat/rord10 /dev/raw/raw10 /dev/sdl1
/tpcc/dat/rord11 /dev/raw/raw11 /dev/sdm1
/tpcc/dat/rord12 /dev/raw/raw12 /dev/sdn1
/tpcc/dat/rord13 /dev/raw/raw13 /dev/sdo1
/tpcc/dat/rord14 /dev/raw/raw14 /dev/sdp1
/tpcc/dat/rord15 /dev/raw/raw15 /dev/sdq1
/tpcc/dat/rord16 /dev/raw/raw16 /dev/sdr1
/tpcc/dat/rord17 /dev/raw/raw17 /dev/sds1
/tpcc/dat/rord18 /dev/raw/raw18 /dev/sdt1
/tpcc/dat/rord19 /dev/raw/raw19 /dev/sdu1
/tpcc/dat/rord20 /dev/raw/raw20 /dev/sdv1
/tpcc/dat/rstock1 /dev/raw/raw21 /dev/sdc2
/tpcc/dat/rstock2 /dev/raw/raw22 /dev/sdd2
/tpcc/dat/rstock3 /dev/raw/raw23 /dev/sde2
/tpcc/dat/rstock4 /dev/raw/raw24 /dev/sdf2
/tpcc/dat/rstock5 /dev/raw/raw25 /dev/sdg2
/tpcc/dat/rstock6 /dev/raw/raw26 /dev/sdh2
/tpcc/dat/rstock7 /dev/raw/raw27 /dev/sdi2
/tpcc/dat/rstock8 /dev/raw/raw28 /dev/sdj2
/tpcc/dat/rstock9 /dev/raw/raw29 /dev/sdk2
/tpcc/dat/rstock10 /dev/raw/raw30 /dev/sdl2
/tpcc/dat/rstock11 /dev/raw/raw31 /dev/sdm2
/tpcc/dat/rstock12 /dev/raw/raw32 /dev/sdn2
/tpcc/dat/rstock13 /dev/raw/raw33 /dev/sdo2
/tpcc/dat/rstock14 /dev/raw/raw34 /dev/sdp2
/tpcc/dat/rstock15 /dev/raw/raw35 /dev/sdq2
/tpcc/dat/rstock16 /dev/raw/raw36 /dev/sdr2
/tpcc/dat/rstock17 /dev/raw/raw37 /dev/sds2
/tpcc/dat/rstock18 /dev/raw/raw38 /dev/sdt2
/tpcc/dat/rstock19 /dev/raw/raw39 /dev/sdu2
/tpcc/dat/rstock20 /dev/raw/raw40 /dev/sdv2
/tpcc/dat/rcust1 /dev/raw/raw41 /dev/sdc3
/tpcc/dat/rcust2 /dev/raw/raw42 /dev/sdd3
/tpcc/dat/rcust3 /dev/raw/raw43 /dev/sde3
/tpcc/dat/rcust4 /dev/raw/raw44 /dev/sdf3
/tpcc/dat/rcust5 /dev/raw/raw45 /dev/sdg3
/tpcc/dat/rcust6 /dev/raw/raw46 /dev/sdh3
/tpcc/dat/rcust7 /dev/raw/raw47 /dev/sdi3
/tpcc/dat/rcust8 /dev/raw/raw48 /dev/sdj3
/tpcc/dat/rcust9 /dev/raw/raw49 /dev/sdk3
/tpcc/dat/rcust10 /dev/raw/raw50 /dev/sdl3
/tpcc/dat/rcust11 /dev/raw/raw51 /dev/sdm3
/tpcc/dat/rcust12 /dev/raw/raw52 /dev/sdn3
/tpcc/dat/rcust13 /dev/raw/raw53 /dev/sdo3
/tpcc/dat/rcust14 /dev/raw/raw54 /dev/sdp3
/tpcc/dat/rcust15 /dev/raw/raw55 /dev/sdq3
/tpcc/dat/rcust16 /dev/raw/raw56 /dev/sdr3
/tpcc/dat/rcust17 /dev/raw/raw57 /dev/sds3
/tpcc/dat/rcust18 /dev/raw/raw58 /dev/sdt3
/tpcc/dat/rcust19 /dev/raw/raw59 /dev/sdu3
/tpcc/dat/rcust20 /dev/raw/raw60 /dev/sdv3
/tpcc/dat/rcustidx1 /dev/raw/raw61 /dev/sdc5
/tpcc/dat/rcustidx2 /dev/raw/raw62 /dev/sdd5
/tpcc/dat/rcustidx3 /dev/raw/raw63 /dev/sde5
/tpcc/dat/rcustidx4 /dev/raw/raw64 /dev/sdf5
/tpcc/dat/rcustidx5 /dev/raw/raw65 /dev/sdg5
/tpcc/dat/rcustidx6 /dev/raw/raw66 /dev/sdh5
/tpcc/dat/rcustidx7 /dev/raw/raw67 /dev/sdi5
/tpcc/dat/rcustidx8 /dev/raw/raw68 /dev/sdj5
/tpcc/dat/rcustidx9 /dev/raw/raw69 /dev/sdk5
/tpcc/dat/rcustidx10 /dev/raw/raw70 /dev/sdl5
/tpcc/dat/rcustidx11 /dev/raw/raw71 /dev/sdm5
/tpcc/dat/rcustidx12 /dev/raw/raw72 /dev/sdn5
/tpcc/dat/rcustidx13 /dev/raw/raw73 /dev/sdo5
/tpcc/dat/rcustidx14 /dev/raw/raw74 /dev/sdp5
/tpcc/dat/rcustidx15 /dev/raw/raw75 /dev/sdq5
/tpcc/dat/rcustidx16 /dev/raw/raw76 /dev/sdr5
/tpcc/dat/rcustidx17 /dev/raw/raw77 /dev/sds5
/tpcc/dat/rcustidx18 /dev/raw/raw78 /dev/sdt5
/tpcc/dat/rcustidx19 /dev/raw/raw79 /dev/sdu5
/tpcc/dat/rcustidx20 /dev/raw/raw80 /dev/sdv5
/tpcc/dat/rordline1 /dev/raw/raw81 /dev/mapper/olvg-ol1
/tpcc/dat/rordline2 /dev/raw/raw82 /dev/mapper/olvg-ol2
/tpcc/dat/rordline3 /dev/raw/raw83 /dev/mapper/olvg-ol3
/tpcc/dat/rordline4 /dev/raw/raw84 /dev/mapper/olvg-ol4
/tpcc/dat/rordline5 /dev/raw/raw85 /dev/mapper/olvg-ol5
/tpcc/dat/rordline6 /dev/raw/raw86 /dev/mapper/olvg-ol6
/tpcc/dat/rordline7 /dev/raw/raw87 /dev/mapper/olvg-ol7
/tpcc/dat/rordline8 /dev/raw/raw88 /dev/mapper/olvg-ol8
/tpcc/dat/rordline9 /dev/raw/raw89 /dev/mapper/olvg-ol9
/tpcc/dat/rordline10 /dev/raw/raw90 /dev/mapper/olvg-ol10
/tpcc/dat/rordline11 /dev/raw/raw91 /dev/mapper/olvg-ol11
/tpcc/dat/rordline12 /dev/raw/raw92 /dev/mapper/olvg-ol12
/tpcc/dat/rordline13 /dev/raw/raw93 /dev/mapper/olvg-ol13
/tpcc/dat/rordline14 /dev/raw/raw94 /dev/mapper/olvg-ol14
/tpcc/dat/rordline15 /dev/raw/raw95 /dev/mapper/olvg-ol15
/tpcc/dat/rordline16 /dev/raw/raw96 /dev/mapper/olvg-ol16
/tpcc/dat/rordline17 /dev/raw/raw97 /dev/mapper/olvg-ol17
/tpcc/dat/rordline18 /dev/raw/raw98 /dev/mapper/olvg-ol18
/tpcc/dat/rordline19 /dev/raw/raw99 /dev/mapper/olvg-ol19
/tpcc/dat/rordline20 /dev/raw/raw100 /dev/mapper/olvg-ol20
/tpcc/dat/master.dat /dev/raw/raw101 /dev/sdc7
/tpcc/dat/def1 /dev/raw/raw102 /dev/sdd7
/tpcc/dat/rware /dev/raw/raw103 /dev/sde7
/tpcc/dat/rdist /dev/raw/raw104 /dev/mapper/olvg-ol21
/tpcc/dat/ritem /dev/raw/raw105 /dev/sdg7
/tpcc/dat/logdisk1 /dev/raw/raw106 /dev/sdw1
/tpcc/dat/logdisk2 /dev/raw/raw107 /dev/sdw2
/tpcc/dat/logdisk3 /dev/raw/raw108 /dev/sdw3
```

sysctl.conf

```
# Kernel sysctl configuration file for Red Hat Linux
#
# For binary values, 0 is disabled, 1 is enabled.  See sysctl(8)
and
# sysctl.conf(5) for more details.

# Controls IP packet forwarding
net.ipv4.ip_forward = 0

# Controls source route verification
net.ipv4.conf.default.rp_filter = 1

# Do not accept source routing
net.ipv4.conf.default.accept_source_route = 0

# Controls the System Request debugging functionality of the
kernel
kernel.sysrq = 0

# Controls whether core dumps will append the PID to the
core filename
# Useful for debugging multi-threaded applications
kernel.core_uses_pid = 1

# Controls the use of TCP syncookies
net.ipv4.tcp_syncookies = 1

# Controls the maximum size of a message, in bytes
kernel.msgmnb = 65536

# Controls the default maxmimum size of a mesage queue
kernel.msgmax = 65536
```

```
kernel.core_uses_pid = 1
kernel.shmmax = 136028291072
kernel.shmall = 136028291072
kernel.shmmni = 4096
kernel.sem = 512 32000 512 256
fs.file-max = 131272
net.ipv4.ip_local_port_range = 1024 65000
net.core.rmem_default=262144
net.core.rmem_max=262144
net.core.wmem_default=262144
net.core.wmem_max=262144
fs.aio-max-nr=1000000
kernel.exec-shield=0
```

set_adapters.sh

```
#!/bin/bash

for adap in host1 host2 host3 host4 host5 host6 host7 host8
do
if [[ $adap = host7 || $adap = host8 ]];then
echo "log host $adap not set"
else
echo 3 > /sys/class/scsi_host/$adap/lpfc_poll
echo 1 > /sys/class/scsi_host/$adap/lpfc_poll_tmo
fi

done

/root/scripts/set_irq.sh

for x in c d e f g h i j k l m n o p q r s t u v
do
echo 512 > /sys/block/sd${x}/queue/nr_requests
done
```

set_irq.sh

```
#!/bin/bash

echo "01" > /proc/irq/33/smp_affinity
echo "02" > /proc/irq/34/smp_affinity
echo "04" > /proc/irq/65/smp_affinity
echo "08" > /proc/irq/66/smp_affinity
echo "10" > /proc/irq/273/smp_affinity
echo "20" > /proc/irq/274/smp_affinity
echo "40" > /proc/irq/289/smp_affinity
echo "80" > /proc/irq/290/smp_affinity
```

12 Appendix - C: Database Setup Code

12.1. Database Creation Scripts

bld_system

```
#!/bin/sh
locktype=${1:-allpages}
partition=10
hist_part=200
export partition hist_part
echo "!!! Using $locktype tables !!!"
echo "!!! you have 5 seconds to abort !!!"
sleep 5
rm errorlog.15_0

setupscript=`pwd`/bm_setup

. $setupscript

verstr=$version; export verstr
if [ $version = "12_0" ]
then
   bmbinary=$SQL_RELEASE/bin/buildmaster.$verstr; export
bmbinary
   install_script=$SQL_RELEASE/scripts/installmaster.$verstr
else
   bmbinary=$SYBASE/$SYBASE_ASE/bin/dataserver ;
export bmbinary
   install_script=$SYBASE/$SYBASE_ASE/scripts/installmast
er; export install_script
fi

if [ "$BINDATASERVER" != "" ]
then
   bmbinary=$BINDATASERVER
fi

tpcc_scripts=$TPCC_DIR/tpcc_scripts; export tpcc_scripts
pagesize=$pagesize; export pagesize

export BLD_SYSTEM=1
#
#
# RCP FROM EARL RELEASE AREA
BUILDMASTER,INSTALLMASTER,DATASERVER
#
#

trap "" 22
echo `date` "Started bld_system for 15_0"

# Build the device.
echo "devcreate.sh buildmaster $bmbinary $verstr
$pagesize"
`devcreate.sh buildmaster $bmbinary $verstr $pagesize <
devices.$version`
sleep 240
# Boot server, run installmaster
run_server - -c./load_$verstr.cfg  &
echo "Install master device"
sleep 240
isql -Usa -P < $install_script > $$_im.log
echo "Finished install master device"
sleep 240
msv_rpc_attach.sh

# Build devices, database, and segments
echo `date` "Creating devices, databases and segments"
devcreate.sh sql System10 < devices.$version | isql -e -Usa -
P
echo `date` " Finished building database"

# Extend tempdb
echo "extending tempdb"
$tpcc_scripts/tpcc_ext_tempdb.sh

shutdown_server.sh
sleep 240

# Start server with no logging  for load.
# Load the data; shutdown again.
./run_server - -c./load_$verstr.cfg -T699 -T3474 &
sleep 240

# Create tables, some indexes, and administrative procs.
echo "Create tables, some indexes, and administrative procs"
$tpcc_scripts/tpcc_tables_parallel$vhash.sh $locktype
$partition $hist_part
echo " Finished Create tables, some indexes, and
administrative procs"

echo "Start $tpcc_scripts/tpcc_admin.sh"
$tpcc_scripts/tpcc_admin.sh
echo "Finished $tpcc_scripts/tpcc_admin.sh"

echo "Start generic_procs.sh "
generic_procs.sh
echo "Finished generic_procs.sh"

# Truncate log, checkpoint, and shutdown
echo "Start Truncate log, checkpoint, and shutdown"
echo "Finished Truncate log, checkpoint, and shutdown"

echo "Start $tpcc_scripts/tpcc_largeio.sh"
$tpcc_scripts/tpcc_largeio.sh
echo "Finished  $tpcc_scripts/tpcc_largeio.sh"

echo `date` " Started loading data"
$tpcc_scripts/tpcc_load_parallel$vhash.sh 1 $DB_SCALE
$partition
echo `date` " Finished loading data"

if [ $vhash = "_vhash" ]
then
   echo `date` " Started unpartitioning of new_order, orders
and order_line tables"
   $tpcc_scripts/tpcc_unpartition.sh
   echo `date` " Finished unpartitioning of new_order, orders
and order_line tables"
fi

shutdown_server.sh

sleep 240

run_server - -c./load_$verstr.cfg -T3474
sleep 240

#
# create indexes, config trips, load stored procs,

echo `date` -- Started building indexes
$tpcc_scripts/tpcc_indexes_parallel$vhash.sh
echo `date` -- Finished building indexes

echo "Start $tpcc_scripts/tpcc_proc.sh"
$tpcc_scripts/tpcc_proc$vhash.sh
echo "Finished $tpcc_scripts/tpcc_proc.sh"
echo "Start $tpcc_scripts/spads_admin.sh"
$tpcc_scripts/spads_admin.sh
echo "Finished $tpcc_scripts/spads_admin.sh"

dumptran_server.sh master
dumptran_server.sh tpcc
```

shutdown_server.sh

```
unset BLD_SYSTEM
cd $WORK_DIR
echo `date` "Finished bld_system"
```

tpcc_admin.sh

```
#!/bin/sh -f

dbname=${1:-tpcc}

#cat << EOF
isql -Usa -P$PASSWORD <<EOF
use $dbname
go


if exists (select * from sysobjects where name = 'tc_infotab'
and type = 'U')
        drop table tc_infotab
go
create table tc_infotab
(
        clients         int     not null,
        duration        int     not null,
        engines         int     not null,
        warehouses      int     not null,
        arrived         int     not null,
        deliveryRatio   int     not null
)
go

if exists (select * from sysobjects where name = 'tc_stattab'
and type = 'U')
        drop table tc_stattab
go
create table tc_stattab
(
        name            char(30)                not null,
        total           int     default 0       not null,
        limit           int     default 0       not null,
        overlimit       int     default 0       not null,
        deadlock        int     default 0       not null,
        sumResp         float   default 0       not null,
        sumRespSq       float   default 0       not null,
        filler1         char(255) default ""    not null,
        filler2         char(255) default ""    not null,
        filler3         char(255) default ""    not null,
        filler4         char(255) default ""    not null,

        unique(name)
)
go

if exists (select * from sysobjects where name = 'tc_synctab'
and type = 'U')
   drop table tc_synctab
go
create table tc_synctab
(
   junk      int    default 0 not null,
)
go

if exists (select * from sysobjects where name = 'tc_initialize'
and type = 'P')
   drop proc tc_initialize
go
create proc tc_initialize
   @clients  int,
   @duration   int = 600,
   @engines   int = 0,
   @warehouses   int = 0,
   @deliveryRatio  int
as
declare @availware int
declare @online      int

   select @availware = count(*) from warehouse

   if (@availware < @warehouses or @warehouses = 0)
      select @warehouses = @availware

   select @online = count(*)
      from master..sysengines
      where status = "online"
   while (@engines > 0 and @engines > @online)
   begin
      dbcc engine("online")
      select @online = @online + 1
   end
   while (@engines > 0 and @engines < @online)
   begin
      dbcc engine(offline)
      select @online = @online + 1
   end

   begin transaction
   delete from tc_infotab
   insert into tc_infotab( clients, arrived,  duration,
                           engines,  warehouses, deliveryRatio)
    values(@clients,       1, @duration, @online, @warehouses,
@deliveryRatio)
   commit transaction

   print "Using %1! warehouses (out of %2!), %3! engines,
and %4! users for %5! seconds",
         @warehouses, @availware, @online, @clients,
@duration

   begin transaction
   delete from tc_synctab
   insert into tc_synctab(junk) values (0)
   commit transaction

   begin transaction
   delete from tc_stattab
   commit transaction

   select engines, warehouses from tc_infotab
go

if exists (select * from sysobjects where name = 'tc_initstat'
and type = 'P')
   drop proc tc_initstat
go
create proc tc_initstat
   @name char(30),
   @limit   int
as
begin
   begin transaction
   insert into tc_stattab(name,limit) values (@name, @limit)
   commit transaction
end
go

if exists (select * from sysobjects where name = 'tc_startup'
and type = 'P')
   drop proc tc_startup
go
create proc tc_startup
as
declare @junk      int
begin
   update tc_infotab set arrived = arrived+1
   select @junk=junk from tc_synctab
   select engines, warehouses from tc_infotab
end
go

if exists (select * from sysobjects where name = 'tc_addstats'
and type = 'P')
   drop proc tc_addstats
go
```

```
create proc tc_addstats
   @name    char(30),
   @total      int,
   @overlimit  int,
   @deadlock int,
   @sumResp float,
   @sumRespSq   float
as
begin
   begin transaction
   update tc_stattab set
      total     = total + @total,
      overlimit = overlimit + @overlimit,
      deadlock  = deadlock  + @deadlock,
      sumResp   = sumResp   + @sumResp,
      sumRespSq = sumRespSq  + @sumRespSq
   where name = @name
   commit transaction
end
go

if exists (select * from sysobjects where name = 'showstats'
and type = 'P')
   drop proc showstats
go
create proc showstats
as
declare @alltotal  float,
   @minutes   float
begin
   select @alltotal = sum(total)
      from tc_stattab
      where not name = "delivery"
   select @minutes = convert(float,duration)/60.0 from
tc_infotab
   select
      convert(char(20), name) as Name,
      convert(decimal(6,0),total/@minutes) as PerMin,
      convert(decimal(6,3),sumResp/(total*1000.0)) as
Average,
      convert(decimal(6,3),
         (sumRespSq-
sumResp*sumResp/total)/(total*1000000.0))
         as Variance,
     convert(decimal(6,3), convert(float, limit)/1000.0) as Limit,
      convert(decimal(6,3),
         (convert(float, overlimit)*100.0)/total) as PercOver,
      convert(decimal(4,0),deadlock) as Deadlocks,
      convert(decimal(5,2),total/@alltotal*100.0) as PercMix
       from tc_stattab
       where  total>0
end
go

if exists (select * from sysobjects where name = 'spread')
   drop view spread
go
create view spread as
select   d_w_id ware,
      d_id dist,
      (select min(no_o_id) from new_order
         where   no_w_id = d.d_w_id
         and no_d_id = d.d_id) first,
   d_next_o_id next
   from district d
go
EOF
```

spads_admin.sh

```
#!/bin/sh -f

dbname=${1:-tpcc}

#cat << EOF
isql -Usa -P$PASSWORD <<EOF
use $dbname
go


/* This table is used to insert TPCC data into SpaDS */
if exists (select * from sysobjects where name =
'spads_summary' and type = 'U')
      drop table spads_summary
go

create table spads_summary
(
      no_permin            decimal(7,3)    null,
      ptid_permin          decimal(7,3)    null,
      ptname_permin        decimal(7,3)    null,
      osid_permin          decimal(7,3)    null,
      osname_permin        decimal(7,3)    null,
      delivery_permin      decimal(7,3)    null,
      stock_permin         decimal(7,3)    null,
      no_response          decimal(7,3)    null,
      ptid_response        decimal(7,3)    null,
      ptname_reponse       decimal(7,3)    null,
      osid_response        decimal(7,3)    null,
      osname_reponse       decimal(7,3)    null,
      delivery_response    decimal(7,3)    null,
      stock_response       decimal(7,3)    null,
      no_variance          decimal(7,3)    null,
      ptid_variance        decimal(7,3)    null,
      ptname_variance      decimal(7,3)    null,
      osid_variance        decimal(7,3)    null,
      osname_variance      decimal(7,3)    null,
      delivery_variance    decimal(7,3)    null,
      stock_variance       decimal(7,3)    null,
      no_limit             decimal(7,3)    null,
      ptid_limit           decimal(7,3)    null,
      ptname_limit         decimal(7,3)    null,
      osid_limit           decimal(7,3)    null,
      osname_limit         decimal(7,3)    null,
      delivery_limit       decimal(7,3)    null,
      stock_limit          decimal(7,3)    null,
      no_prcover           decimal(7,3)    null,
      ptid_prcover         decimal(7,3)    null,
      ptname_prcover       decimal(7,3)    null,
      osid_prcover         decimal(7,3)    null,
      osname_prcover       decimal(7,3)    null,
      delivery_prcover     decimal(7,3)    null,
      stock_prcover        decimal(7,3)    null,
      no_deadlock          decimal(7,3)    null,
      ptid_deadlock        decimal(7,3)    null,
      ptname_deadlock      decimal(7,3)    null,
      osid_deadlock        decimal(7,3)    null,
      osname_deadlock      decimal(7,3)    null,
      delivery_deadlock    decimal(7,3)    null,
      stock_deadlock       decimal(7,3)    null,
      no_prcmix            decimal(7,3)    null,
      ptid_prcmix          decimal(7,3)    null,
    ptname_prcmix          decimal(7,3)    null,
      osid_prcmix          decimal(7,3)    null,
      osname_prcmix        decimal(7,3)    null,
      delivery_prcmix      decimal(7,3)    null,
      stock_prcmix         decimal(7,3)    null

)
go

if exists (select * from sysobjects where name =
'insert_summary' and type = 'P')
      drop proc insert_summary
go

create proc insert_summary
as

      declare @no_permin            decimal(7,3)
      declare @ptid_permin          decimal(7,3)
      declare @ptname_permin        decimal(7,3)
      declare @osid_permin          decimal(7,3)
      declare @osname_permin        decimal(7,3)
      declare @delivery_permin      decimal(7,3)
      declare @stock_permin         decimal(7,3)
```

```
    declare @no_response            decimal(7,3)
    declare @ptid_response          decimal(7,3)
    declare @ptname_response        decimal(7,3)
    declare @osid_response          decimal(7,3)
    declare @osname_response        decimal(7,3)
    declare @delivery_response      decimal(7,3)
    declare @stock_response         decimal(7,3)
    declare @no_variance            decimal(7,3)
    declare @ptid_variance          decimal(7,3)
    declare @ptname_variance        decimal(7,3)
    declare @osid_variance          decimal(7,3)
    declare @osname_variance        decimal(7,3)
    declare @delivery_variance      decimal(7,3)
    declare @stock_variance         decimal(7,3)
    declare @no_limit               decimal(7,3)
    declare @ptid_limit             decimal(7,3)
    declare @ptname_limit           decimal(7,3)
    declare @osid_limit             decimal(7,3)
    declare @osname_limit           decimal(7,3)
    declare @delivery_limit         decimal(7,3)
    declare @stock_limit            decimal(7,3)
    declare @no_prcover             decimal(7,3)
    declare @ptid_prcover           decimal(7,3)
    declare @ptname_prcover         decimal(7,3)
    declare @osid_prcover           decimal(7,3)
    declare @osname_prcover         decimal(7,3)
    declare @delivery_prcover       decimal(7,3)
    declare @stock_prcover          decimal(7,3)
    declare @no_deadlock            decimal(7,3)
    declare @ptid_deadlock          decimal(7,3)
    declare @ptname_deadlock        decimal(7,3)
    declare @osid_deadlock          decimal(7,3)
    declare @osname_deadlock        decimal(7,3)
    declare @delivery_deadlock      decimal(7,3)
    declare @stock_deadlock         decimal(7,3)
    declare @no_prcmix              decimal(7,3)
    declare @ptid_prcmix            decimal(7,3)
    declare @ptname_prcmix          decimal(7,3)
  declare @osid_prcmix            decimal(7,3)
    declare @osname_prcmix          decimal(7,3)
    declare @delivery_prcmix        decimal(7,3)
    declare @stock_prcmix           decimal(7,3)

    declare @alltotal               float
    declare @minutes                float


begin


  select @alltotal = sum(total)
    from tc_stattab
    where not name = "delivery"

    select @minutes = convert(float,duration)/60.0 from
tc_infotab



    /* new_order */
    select
@no_permin=convert(decimal(6,0),total/@minutes),

@no_response=convert(decimal(6,3),sumResp/(total*1000.0)
),
            @no_variance=convert(decimal(6,3),
                  (sumRespSq-
sumResp*sumResp/total)/(total*1000000.0)),
            @no_limit=convert(decimal(6,3), convert(float,
limit)/1000,0),
            @no_prcover=convert(decimal(6,3),
                        (convert(float, overlimit)*100.0)/total),
            @no_deadlock=convert(decimal(4,0),deadlock),

@no_prcmix=convert(decimal(5,2),total/@alltotal*100.0)
            from tc_stattab
            where name="new_order"
            and total > 0

    /* payment_byid */
    select
@ptid_permin=convert(decimal(6,0),total/@minutes),

@ptid_response=convert(decimal(6,3),sumResp/(total*1000.
0)),
            @ptid_variance=convert(decimal(6,3),
                        (sumRespSq-
sumResp*sumResp/total)/(total*1000000.0)),
            @ptid_limit=convert(decimal(6,3), convert(float,
limit)/1000,0),
            @ptid_prcover=convert(decimal(6,3),
                        (convert(float, overlimit)*100.0)/total),
            @ptid_deadlock=convert(decimal(4,0),deadlock),

@ptid_prcmix=convert(decimal(5,2),total/@alltotal*100.0)
            from tc_stattab
            where name="payment_byid"
            and total > 0

    /* payment_byname */
    select
@ptname_permin=convert(decimal(6,0),total/@minutes),

@ptname_response=convert(decimal(6,3),sumResp/(total*10
00.0)),
            @ptname_variance=convert(decimal(6,3),
                        (sumRespSq-
sumResp*sumResp/total)/(total*1000000.0)),
            @ptname_limit=convert(decimal(6,3), convert(float,
limit)/1000,0),
            @ptname_prcover=convert(decimal(6,3),
                        (convert(float, overlimit)*100.0)/total),

@ptname_deadlock=convert(decimal(4,0),deadlock),

@ptname_prcmix=convert(decimal(5,2),total/@alltotal*100.0)
            from tc_stattab
            where name="payment_byname"
            and total > 0

  /* order_status_byid */
    select
@osid_permin=convert(decimal(6,0),total/@minutes),

@osid_response=convert(decimal(6,3),sumResp/(total*1000.
0)),
            @osid_variance=convert(decimal(6,3),
                        (sumRespSq-
sumResp*sumResp/total)/(total*1000000.0)),
            @osid_limit=convert(decimal(6,3), convert(float,
limit)/1000,0),
            @osid_prcover=convert(decimal(6,3),
                        (convert(float, overlimit)*100.0)/total),
            @osid_deadlock=convert(decimal(4,0),deadlock),

@osid_prcmix=convert(decimal(5,2),total/@alltotal*100.0)
            from tc_stattab
            where name="order_status_byid"
            and total > 0



    /* order_status byname */
    select
@osname_permin=convert(decimal(6,0),total/@minutes),

@osname_response=convert(decimal(6,3),sumResp/(total*1
000.0)),
            @osname_variance=convert(decimal(6,3),
                        (sumRespSq-
sumResp*sumResp/total)/(total*1000000.0)),
            @osname_limit=convert(decimal(6,3),  convert(float,
limit)/1000,0),
            @osname_prcover=convert(decimal(6,3),
                        (convert(float, overlimit)*100.0)/total),

@osname_deadlock=convert(decimal(4,0),deadlock),
```

```
@osname_prcmix=convert(decimal(5,2),total/@alltotal*100.0)
        from tc_stattab
        where name="order_status_byname"
        and total > 0

    /* delivery */
    select
@delivery_permin=convert(decimal(6,0),total/@minutes),

@delivery_response=convert(decimal(6,3),sumResp/(total*1
000.0)),
        @delivery_variance=convert(decimal(6,3),
                (sumRespSq-
sumResp*sumResp/total)/(total*1000000.0)),
        @delivery_limit=convert(decimal(6,3), convert(float,
limit)/1000,0),
        @delivery_prcover=convert(decimal(6,3),
                (convert(float, overlimit)*100.0)/total),

@delivery_deadlock=convert(decimal(4,0),deadlock),

@delivery_prcmix=convert(decimal(5,2),total/@alltotal*100.0)
        from tc_stattab
        where name="delivery"
        and total > 0

  /* stock_level */
    select
@stock_permin=convert(decimal(6,0),total/@minutes),

@stock_response=convert(decimal(6,3),sumResp/(total*100
0.0)),
        @stock_variance=convert(decimal(6,3),
                (sumRespSq-
sumResp*sumResp/total)/(total*1000000.0)),
        @stock_limit=convert(decimal(6,3), convert(float,
limit)/1000,0),
        @stock_prcover=convert(decimal(6,3),
                (convert(float, overlimit)*100.0)/total),
        @stock_deadlock=convert(decimal(4,0),deadlock),

@stock_prcmix=convert(decimal(5,2),total/@alltotal*100.0)
        from tc_stattab
        where name="stock_level"
        and total > 0


  insert into spads_summary values(
    @no_permin,
    @ptid_permin,
    @ptname_permin,
    @osid_permin,
    @osname_permin,
    @delivery_permin,
    @stock_permin,
    @no_response,
    @ptid_response,
    @ptname_response,
    @osid_response,
    @osname_response,
    @delivery_response,
    @stock_response,
    @no_variance,
    @ptid_variance,
    @ptname_variance,
    @osid_variance,
    @osname_variance,
    @delivery_variance,
    @stock_variance,
    @no_limit,
    @ptid_limit,
    @ptname_limit,
    @osid_limit,
    @osname_limit,
    @delivery_limit,
    @stock_limit,
    @no_prcover,
    @ptid_prcover,
    @ptname_prcover,
    @osid_prcover,
    @osname_prcover,
    @delivery_prcover,
    @stock_prcover,
    @no_deadlock,
    @ptid_deadlock,
    @ptname_deadlock,
    @osid_deadlock,
    @osname_deadlock,
    @delivery_deadlock,
    @stock_deadlock,
    @no_prcmix,
    @ptid_prcmix,
    @ptname_prcmix,
    @osid_prcmix,
    @osname_prcmix,
    @delivery_prcmix,
    @stock_prcmix

)

end

go
```

```
if exists (select * from sysobjects where name = 'tc_synctab'
and type = 'U')
  drop table tc_synctab
go
create table tc_synctab
(
  junk     int   default 0 not null,
)
go

if exists (select * from sysobjects where name = 'tc_initialize'
and type = 'P')
  drop proc tc_initialize
go
create proc tc_initialize
  @clients  int,
  @duration   int = 600,
  @engines   int = 0,
  @warehouses   int = 0,
  @deliveryRatio int
as
declare @availware int
declare @online     int

  select @availware = count(*) from warehouse

  if (@availware < @warehouses or @warehouses = 0)
    select @warehouses = @availware

  select @online = count(*)
    from master..sysengines
    where status = "online"
  while (@engines > 0 and @engines > @online)
  begin
    dbcc engine("online")
    select @online = @online + 1
  end
  while (@engines > 0 and @engines < @online)
  begin
    dbcc engine(offline)
    select @online = @online + 1
  end

  begin transaction
  delete from tc_infotab
  insert into tc_infotab( clients, arrived,  duration,
                engines,  warehouses, deliveryRatio)
   values(@clients,      1, @duration, @online, @warehouses,
@deliveryRatio)
  commit transaction
```

```
   print "Using %1! warehouses (out of %2!), %3! engines,
and %4! users for %5! seconds",
         @warehouses, @availware, @online, @clients,
@duration

   begin transaction
   delete from tc_synctab
   insert into tc_synctab(junk) values (0)
   commit transaction

   begin transaction
   delete from tc_stattab
   delete from spads_summary
   commit transaction

   select engines, warehouses from tc_infotab
go

EOF
```

devices.15_0

```
DEVICE master /tpcc/dat/master.dat 500
segment=default segment=system
DEVICE_END
DEVICE fdef1 /tpcc/dat/def1  200
db=tpcc size=200
segment=default segment=system
DEVICE_END
DEVICE fStock1 /tpcc/dat/rstock1 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock2 /tpcc/dat/rstock2 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock3 /tpcc/dat/rstock3 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock4 /tpcc/dat/rstock4 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock5 /tpcc/dat/rstock5 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock6 /tpcc/dat/rstock6 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock7 /tpcc/dat/rstock7 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock8 /tpcc/dat/rstock8 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock9 /tpcc/dat/rstock9 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock10 /tpcc/dat/rstock10 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock11 /tpcc/dat/rstock11 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock12 /tpcc/dat/rstock12 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock13 /tpcc/dat/rstock13 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock14 /tpcc/dat/rstock14 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock15 /tpcc/dat/rstock15 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock16 /tpcc/dat/rstock16 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock17 /tpcc/dat/rstock17 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock18 /tpcc/dat/rstock18 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock19 /tpcc/dat/rstock19 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fStock20 /tpcc/dat/rstock20 44000
db=tpcc size=44000 slices=10
segment=Sstock
DEVICE_END
DEVICE fcust1 /tpcc/dat/rcust1 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust2 /tpcc/dat/rcust2 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust3 /tpcc/dat/rcust3 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust4 /tpcc/dat/rcust4 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust5 /tpcc/dat/rcust5 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust6 /tpcc/dat/rcust6 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust7 /tpcc/dat/rcust7 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust8 /tpcc/dat/rcust8 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust9 /tpcc/dat/rcust9 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust10 /tpcc/dat/rcust10 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust11 /tpcc/dat/rcust11 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust12 /tpcc/dat/rcust12 36000
```

```
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust13 /tpcc/dat/rcust13 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust14 /tpcc/dat/rcust14 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust15 /tpcc/dat/rcust15 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust16 /tpcc/dat/rcust16 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust17 /tpcc/dat/rcust17 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust18 /tpcc/dat/rcust18 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust19 /tpcc/dat/rcust19 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcust20 /tpcc/dat/rcust20 36000
db=tpcc size=36000 slices=10
segment=Scustomer
DEVICE_END
DEVICE fcustidx1 /tpcc/dat/rcustidx1 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx2 /tpcc/dat/rcustidx2 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx3 /tpcc/dat/rcustidx3 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx4 /tpcc/dat/rcustidx4 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx5 /tpcc/dat/rcustidx5 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx6 /tpcc/dat/rcustidx6 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx7 /tpcc/dat/rcustidx7 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx8 /tpcc/dat/rcustidx8 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx9 /tpcc/dat/rcustidx9 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx10 /tpcc/dat/rcustidx10 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx11 /tpcc/dat/rcustidx11 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx12 /tpcc/dat/rcustidx12 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx13 /tpcc/dat/rcustidx13 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx14 /tpcc/dat/rcustidx14 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx15 /tpcc/dat/rcustidx15 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx16 /tpcc/dat/rcustidx16 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx17 /tpcc/dat/rcustidx17 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx18 /tpcc/dat/rcustidx18 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx19 /tpcc/dat/rcustidx19 2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE fcustidx20 /tpcc/dat/rcustidx20  2000
db=tpcc size=1990
segment=Scustomer_index
DEVICE_END
DEVICE rordline1 /tpcc/dat/rordline1 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline2 /tpcc/dat/rordline2 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline3 /tpcc/dat/rordline3 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline4 /tpcc/dat/rordline4 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline5 /tpcc/dat/rordline5 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline6 /tpcc/dat/rordline6 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline7 /tpcc/dat/rordline7 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline8 /tpcc/dat/rordline8 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline9 /tpcc/dat/rordline9 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline10 /tpcc/dat/rordline10 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline11 /tpcc/dat/rordline11 30000
```

```
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline12 /tpcc/dat/rordline12 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline13 /tpcc/dat/rordline13 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline14 /tpcc/dat/rordline14 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline15 /tpcc/dat/rordline15 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline16 /tpcc/dat/rordline16 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline17 /tpcc/dat/rordline17 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline18 /tpcc/dat/rordline18 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline19 /tpcc/dat/rordline19 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rordline20 /tpcc/dat/rordline20 30000
db=tpcc size=30000 slices=10
segment=Sorder_line
DEVICE_END
DEVICE rord1 /tpcc/dat/rord1 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord2 /tpcc/dat/rord2 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord3 /tpcc/dat/rord3 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord4 /tpcc/dat/rord4 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord5 /tpcc/dat/rord5 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord6 /tpcc/dat/rord6 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord7 /tpcc/dat/rord7 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord8 /tpcc/dat/rord8 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord9 /tpcc/dat/rord9 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord10 /tpcc/dat/rord10 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord11 /tpcc/dat/rord11 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord12 /tpcc/dat/rord12 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord13 /tpcc/dat/rord13 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord14 /tpcc/dat/rord14 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord15 /tpcc/dat/rord15 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord16 /tpcc/dat/rord16 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord17 /tpcc/dat/rord17 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord18 /tpcc/dat/rord18 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord19 /tpcc/dat/rord19 5600
db=tpcc size=3600
segment=Scache
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rord20 /tpcc/dat/rord20 5600
db=tpcc size=3600
segment=Scache
```

```
segment=Sorders
segment=Shistory
DEVICE_END
DEVICE rware /tpcc/dat/rware 128
db=tpcc size=128
segment=Sware
DEVICE_END
DEVICE rdist /tpcc/dat/rdist 1260
db=tpcc size=1260
segment=Sdist
DEVICE_END
DEVICE ritem /tpcc/dat/ritem 16
db=tpcc size=16
segment=Sitem
DEVICE_END
DEVICE logdisk1 /tpcc/dat/logdisk1 25000
db=tpcc size=25000 log
DEVICE_END
DEVICE logdisk2 /tpcc/dat/logdisk2 180000
db=tpcc size=180000 log
DEVICE_END
DEVICE logdisk3 /tpcc/dat/logdisk3 180000
db=tpcc size=180000 log
DEVICE_END
```

tpcc_ext_tempdb.sh

```
#!/bin/sh -x

isql -Usa -P << EOF
alter database tempdb on master=100
go
EOF
```

generic_procs.sh

```
#!/bin/sh -f
isql -Usa -P$PASSWORD <<EOF
use ${1:-tpcc}
go

if exists (select * from sysobjects where name = "indexes"
and type = 'P')
   drop proc indexes
go
if exists (select * from sysobjects where name = "rowsize"
and type = 'P')
   drop proc rowsize
go
if exists (select * from sysobjects where name = "spaceused"
and type = 'P')
   drop proc spaceused
go
if exists (select * from sysobjects where name = "columns"
and type = 'P')
   drop proc columns
go

create proc indexes as
   select    name, indid, ipgtrips, status2
      from   sysindexes
      where   id > 1000
         and indid > 0
go

create proc rowsize @table_name varchar(30) = NULL as
   select
      o.name,
      bytes = sum(c.length),
      rowsPerPage = 2016/(sum(c.length)+4)
   from
      sysobjects o, syscolumns c
   where
      (@table_name = NULL OR o.name = @table_name)
AND
      o.type = 'U' AND
      o.id>999 AND
      o.id = c.id
   group by
      o.id
go

create proc spaceused as
declare @table_namechar(30)
declare @eot_so     int
declare c_msu_so cursor for
   select name
      from sysobjects
      where   id>9999 and
         type = 'U'
begin
   open c_msu_so
   select @eot_so = 0
   while (@eot_so = 0) begin
      fetch c_msu_so into @table_name
      if (@@sqlstatus > 0) begin
         select @eot_so = 1
         break
      end
      exec sp_spaceused @table_name
   end
end
go

create proc columns @table_name varchar(30) as
   select
      c.name, c.length, c.type, c.prec, c.scale
   from
      sysobjects o, syscolumns c
   where
      o.name = @table_name AND
      o.type = 'U' AND
      o.id = c.id
go
EOF
```

tpcc_largeio.sh

```
#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF
dbcc iosize("tpcc", "stock", 16)
go
dbcc iosize("tpcc", "customer", 16)
go
dbcc iosize("tpcc", "order_line", 16)
go
EOF
```

tpcc_tables_parallel_vhash.sh

```
#!/bin/ksh -x

wh=$DB_SCALE
wh_p10=$((wh+(wh+9)/10))

partition=${2:-1}
hist_part=${3:-1}

isql -Usa -P$PASSWORD -e << EOF
/* This script will create all the tables required for TPC-C
benchmark */
/* It will also create some of the indexes. */
sp_dboption tpcc,"select into/bulkcopy",true
go
use tpcc
go
checkpoint
go
```

```
if exists ( select name from sysobjects where name =
'warehouse' )
   drop table warehouse
go
create table warehouse (
   w_id      smallint,
   w_name      char(10),
   w_street_1  char(20),
   w_street_2  char(20),
   w_city      char(20),
   w_state    char(2),
   w_zip    char(9),
   w_tax    real,
   w_ytd    float      /*- Updated by PID, PNM */
   , constraint w_clu primary key using clustered (w_id) = (41)
      with max $((wh_p10*41+41)) key
) on Sware
go

if exists ( select name from sysobjects where name =
'district' )
   drop table district
go
create table district (
   d_id      tinyint,
   d_w_id     smallint,
   d_name      char(10),
   d_street_1  char(20),
   d_street_2  char(20),
   d_city    char(20),
   d_state    char(2),
   d_zip    char(9),
   d_tax    real,
   d_ytd    float,     /*- Updated by PID, PNM */
   d_next_o_id   int      /*- Updated by NO */
  , constraint d_clu primary key using clustered (d_w_id, d_id)
= (440,40)
      with max $((wh_p10*440+480)) key
) on Sdist
go

if exists ( select name from sysobjects where name =
'customer' )
   drop table customer
go
create table customer (
   c_id      int,
   c_d_id      tinyint,
   c_w_id     smallint,
   c_first    char(16),
   c_middle  char(2),
   c_last    char(16),
   c_street_1  char(20),
   c_street_2  char(20),
   c_city    char(20),
   c_state     char(2),
   c_zip     char(9),
   c_phone      char(16),
   c_since    datetime,
   c_creditchar(2),
   c_credit_limnumeric(12,0),
   c_discount  real,
   c_delivery_cnt   smallint,
   c_payment_cnt smallint,   /*- Updated by PNM, PID */
   c_balance   float,     /*- Updated by PNM, PID */
   c_ytd_payment float,     /*- Updated by PNM, PID */
   c_data1    char(250),   /*- Updated (?) by PNM, PID */
   c_data2    char(250) /*- Updated (?) by PNM, PID */
   , constraint c_clu primary key using clustered (c_w_id, c_id,
c_d_id)
      = (36300, 11, 1)
      with max $((wh_p10*36300+36312)) key
) on Scustomer
go

if exists ( select name from sysobjects where name =
'history' )
   drop table history
go
create table history (
   h_c_id      int,
   h_c_d_id  tinyint,
   h_c_w_id smallint,
   h_d_id     tinyint,
   h_w_id     smallint,
   h_date     datetime,
   h_amountfloat,
   h_data     char(24)
) on Shistory
go
alter table history partition $hist_part
go

if exists ( select name from sysobjects where name =
'new_order' )
   drop table new_order
go
create table new_order (
   no_o_id   int,
   no_d_id   tinyint,
   no_w_id      smallint,
) on Scache
go
declare @dbid   int
select @dbid = db_id("tpcc")
dbcc tune("forload", @dbid, new_order)
alter table new_order partition $partition
go


if exists ( select name from sysobjects where name = 'orders' )
   drop table orders
go
create table orders (
   o_id      int,
   o_c_id      int,
   o_d_id      tinyint,
   o_w_id     smallint,
   o_entry_ddatetime,
   o_carrier_idsmallint,   /*- Updated by D */
   o_ol_cnt   tinyint,
   o_all_local   tinyint
) on Sorders
go


declare @dbid   int
select @dbid = db_id("tpcc")
dbcc tune("forload", @dbid, orders)

alter table orders partition $partition
go

if exists ( select name from sysobjects where name =
'order_line' )
   drop table order_line
go
create table order_line (
   ol_o_id     int,
   ol_d_id     tinyint,
   ol_w_id     smallint,
   ol_number   tinyint,
   ol_i_id      int,
   ol_supply_w_id smallint,
   ol_delivery_d datetime, /*- Updated by D */
   ol_quantity  smallint,
   ol_amount   float,
   ol_dist_info char(24)
) on Sorder_line
go
declare @dbid   int
select @dbid = db_id("tpcc")
dbcc tune("forload", @dbid, order_line)
alter table order_line partition $partition
go
```

```
if exists ( select name from sysobjects where name = 'item' )
   drop table item
go
create table item (
   i_id    int,
   i_im_id     int,
   i_name     char(24),
   i_price     float,
   i_data      char(50)
   , constraint i_clu primary key using clustered (i_id)  = (1)
      with max 110001 key
) on Sitem
go


if exists ( select name from sysobjects where name = 'stock' )
   drop table stock
go
create table stock (
   s_i_id   int,
   s_w_id     smallint,
   s_quantity   smallint,   /*- Updated by NO */
   s_ytd    int,     /*- Updated by NO */
   s_order_cnt smallint,   /*- Updated by NO */
   s_remote_cnt smallint,   /*- Updated by NO */
   s_dist_01 char(24),
   s_dist_02 char(24),
   s_dist_03 char(24),
   s_dist_04 char(24),
   s_dist_05 char(24),
   s_dist_06 char(24),
   s_dist_07 char(24),
   s_dist_08 char(24),
   s_dist_09 char(24),
   s_dist_10 char(24),
   s_data     char(50)
   , constraint s_clu primary key using clustered (s_i_id,
s_w_id)
      = ($wh_p10,1)
      with max $((wh_p10*110000+wh_p10+1)) key
) on Sstock
go

checkpoint
go
EOF
```

tpcc_indexes_parallel_vhash.sh

```
#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF &
/* This script will create the TPC-C indexes that are best
   created after the load. */
use tpcc
go

--alter table customer unpartition
go
/*
create unique clustered index c_clu
   on customer(c_w_id, c_id, c_d_id)
   with sorted_data
   on Scustomer
*/
go

create unique nonclustered index c_non1
       on customer(c_w_id, c_d_id, c_last, c_first, c_id) with
consumers = 3
       on Scustomer_index
go
sp_spaceused customer, 1
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go

--alter table new_order unpartition
go
create unique clustered index no_clu
   on new_order(no_w_id, no_d_id, no_o_id)
   with sorted_data
   on Scache
go
dbcc tune(ascinserts, 1, new_order)
go
dbcc tune(oamtrips, 100, new_order)
go
sp_spaceused new_order, 1
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go

--alter table orders unpartition
go
create unique clustered index o_clu
   on orders(o_w_id, o_d_id, o_id)
   with sorted_data
   on Sorders
go
dbcc tune(ascinserts, 1, orders)
go
dbcc tune(oamtrips, 100, orders)
go
sp_spaceused orders, 1
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go

--alter table order_line unpartition
go
create unique clustered index ol_clu
   on order_line(ol_w_id, ol_d_id, ol_o_id, ol_number)
   with sorted_data
   on Sorder_line
go
dbcc tune(ascinserts, 1, order_line)
go
dbcc tune(oamtrips, 100, order_line)
go
sp_spaceused order_line, 1
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go

--alter table item unpartition
go
/*
create unique clustered index i_clu
   on item(i_id)
   with sorted_data
   on Scache
*/
go
sp_spaceused item, 1
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go
```

```
--alter table stock unpartition
go
/*
create unique clustered index s_clu
   on stock(s_i_id, s_w_id)
   with sorted_data
   on Sstock
*/
go

sp_spaceused stock, 1
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go

/*
create unique clustered index w_clu
       on warehouse(w_id)
       with fillfactor = 1
       on Scache
*/
go
sp_spaceused warehouse, 1
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go
/*
create unique clustered index d_clu
       on district(d_w_id, d_id)
       with fillfactor = 1
       on Scache
*/
go
sp_spaceused district, 1
go
EOF

wait

isql -Usa -P$PASSWORD << EOF &
use tpcc
go
checkpoint
go
EOF
```

```
isql -Usa -P$PASSWORD << EOF &
use tpcc
go
checkpoint
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go
checkpoint
go
EOF

wait
```

tpcc_load_parallel_vhash

```
#!/bin/sh -f
partition=${3:--1}
hist_part=${4:--1}

/tpcc/products/tpcc11/loader/load.$version stock $1 $2
"$PASSWORD" $partition "no_partition" &
/tpcc/products/tpcc11/loader/load.$version customer $1 $2
"$PASSWORD" $partition "no_partition" &
/tpcc/products/tpcc11/loader/load.$version new_order $1 $2
"$PASSWORD" $partition &
/tpcc/products/tpcc11/loader/load.$version warehouse $1 $2
"$PASSWORD"  &
/tpcc/products/tpcc11/loader/load.$version district $1 $2
"$PASSWORD"  &
/tpcc/products/tpcc11/loader/load.$version item 1 1
"$PASSWORD"  $partition "no_partition" &
/tpcc/products/tpcc11/loader/load.$version orders $1 $2
"$PASSWORD" $partition &
/tpcc/products/tpcc11/loader/load.$version history $1 $2
"$PASSWORD" $hist_part &
wait
exit 0
```

tpcc_proc.sh

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

pagesize=8
# Stored procedure for TPC-C 3.2 on SQL Server 11.1 and
later
# Copyright Sybase 1997
#
isql -Usa -P$PASSWORD <<EOF
use tpcc
go
if exists ( SELECT name FROM sysobjects WHERE name =
'neworder_local')
   DROP PROC neworder_local
go

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

   @i_id int = 0, @ol_qty smallint = 0,
   @i_id2  int = 0, @ol_qty2  smallint = 0,
```

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

declare
   @w_tax      real,     @d_tax    real,
   @c_last     char(16), @c_credit   char(2),
   @c_discount real,     @commit_flag   int,
   @c_ins_id   int,    @local_d_id   int,

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

   @s_quantity   smallint,   @ten_smallint   smallint,
   @s_ytd    int,    @s_order_cntint,
   @s_dist     char(24), @s_data     char(50),
   @one_smallint  smallint,   @zero_smallint smallint,
   @ninenine_smallint    smallint,

   @ol_number  int,    @o_id      int,
   @o_entry_d   datetime, @b_g    char(1),
   @ol_amount  float



begin

   begin transaction NO


   -- @#@# UPDATE district FROM district, warehouse,
customer
   --

   UPDATE district
   SET    d_next_o_id = d_next_o_id + 1
     , @o_id   = d_next_o_id
     , @d_tax  = d_tax
     , @commit_flag = 1
     , @ol_number= 0
     , @local_d_id = @d_id
     , @ten_smallint = 10
     , @zero_smallint = 0
     , @ninenine_smallint  = 99
     , @one_smallint= 1
     , @o_entry_d = getdate()
   WHERE     d_w_id  = @w_id
     AND  d_id   = @d_id

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

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

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

        if (@@rowcount = 0)
          begin
            select @commit_flag = 0
            select NULL, NULL, NULL, NULL
            continue
          end
        /*Otherwise if the item is found */
          update stock
             set s_ytd = s_ytd + @ol_qty,
           @ol_amount = @ol_qty * @i_price,
               @s_quantity = s_quantity - @ol_qty +
                     case when (s_quantity - @ol_qty <
@ten_smallint)
                     then @ninenine_smallint else
@zero_smallint end,
                   s_quantity = s_quantity - @ol_qty +
                     case when (s_quantity - @ol_qty <
@ten_smallint)
                     then @ninenine_smallint  else
@zero_smallint end,
               s_order_cnt = s_order_cnt + @one_smallint,
                   @s_data = s_data,
                   @s_dist = case @local_d_id
                        when 1 then s_dist_01
                        when 2 then s_dist_02
                        when 3 then s_dist_03
                        when 4 then s_dist_04
                        when 5 then s_dist_05
                        when 6 then s_dist_06
                        when 7 then s_dist_07
                        when 8 then s_dist_08
                        when 9 then s_dist_09
                        when 10 then s_dist_10
                        end
             where s_w_id = @w_id and
                    s_i_id = @i_id
          if (@@rowcount = 0)
             begin
               select @commit_flag = 0
               select NULL, NULL, NULL, NULL
               continue
          end
           /*Otherwise if the Stock is found */

     INSERT INTO order_line (
        ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
```

```
        ol_supply_w_id, ol_delivery_d, ol_quantity,
        ol_amount, ol_dist_info)
    VALUES (
        @o_id, @d_id, @w_id, @ol_number, @i_id,
        @w_id, "19000101", @ol_qty,
        @ol_amount, @s_dist)


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

    end /* while */

    SELECT   @c_last  = c_last,
       @c_discount    = c_discount,
       @c_credit      = c_credit,
       @c_ins_id  = c_id
    FROM   customer HOLDLOCK
    WHERE   c_w_id  = @w_id
      ANDc_d_id  = @d_id
      ANDc_id   = @c_id

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


    SELECT  @w_tax   = w_tax
    FROM warehouse HOLDLOCK
    WHERE w_id = @w_id



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

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

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

   @i_idint = 0, @s_w_id  smallint = 0, @ol_qty  smallint = 0,
   @i_id2  int = 0, @s_w_id2smallint = 0, @ol_qty2smallint =
0,
   @i_id3  int = 0, @s_w_id3smallint = 0, @ol_qty3smallint =
0,
   @i_id4  int = 0, @s_w_id4smallint = 0, @ol_qty4smallint =
0,
   @i_id5  int = 0, @s_w_id5smallint = 0, @ol_qty5smallint =
0,
   @i_id6  int = 0, @s_w_id6smallint = 0, @ol_qty6smallint =
0,
   @i_id7  int = 0, @s_w_id7smallint = 0, @ol_qty7smallint =
0,
   @i_id8  int = 0, @s_w_id8smallint = 0, @ol_qty8smallint =
0,
   @i_id9  int = 0, @s_w_id9smallint = 0, @ol_qty9smallint =
0,
   @i_id10   int = 0, @s_w_id10  smallint = 0, @ol_qty10
   smallint = 0,
   @i_id11   int = 0, @s_w_id11  smallint = 0, @ol_qty11
   smallint = 0,
   @i_id12   int = 0, @s_w_id12  smallint = 0, @ol_qty12
   smallint = 0,
   @i_id13   int = 0, @s_w_id13  smallint = 0, @ol_qty13
   smallint = 0,
   @i_id14   int = 0, @s_w_id14  smallint = 0, @ol_qty14
   smallint = 0,
   @i_id15   int = 0, @s_w_id15  smallint = 0, @ol_qty15
   smallint = 0
)
as

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

   @s_quantity   smallint,   @ten_smallint   smallint,
   @s_ytd    int,     @s_order_cntint,
   @s_dist      char(24),  @s_data     char(50),
   @one_smallint  smallint,   @zero_smallint smallint,
   @ninenine_smallint    smallint,

   @ol_number  int,     @o_id      int,
   @o_entry_d   datetime, @b_g    char(1),
   @ol_amount  float
begin

   begin transaction NO


   -- @#@# UPDATE district FROM district, warehouse,
customer
   --

   UPDATE district
   SET     d_next_o_id = d_next_o_id + 1
     , @o_id    = d_next_o_id
     , @d_tax  = d_tax
     , @commit_flag = 1
     , @ol_number= 0
     , @local_d_id = @d_id
     , @ten_smallint = 10
     , @zero_smallint = 0
     , @ninenine_smallint  = 99
     , @one_smallint= 1
     , @o_entry_d = getdate()
   WHERE      d_w_id  = @w_id
     AND  d_id   = @d_id

   while (@ol_number < @o_ol_cnt) begin
     SELECT @ol_number = @ol_number + 1
     ,@i_id = case @ol_number
         when 1 then @i_id2
          when 2 then @i_id3
          when 3 then @i_id4
          when 4 then @i_id5
          when 5 then @i_id6
          when 6 then @i_id7
          when 7 then @i_id8
          when 8 then @i_id9
          when 9 then @i_id10
          when 10 then @i_id11
          when 11 then @i_id12
          when 12 then @i_id13
          when 13 then @i_id14
```

```
        when 14 then @i_id15
        else @i_id
        end
    , @ol_qty = case @ol_number
        when 1 then @ol_qty2
        when 2 then @ol_qty3
        when 3 then @ol_qty4
        when 4 then @ol_qty5
        when 5 then @ol_qty6
        when 6 then @ol_qty7
        when 7 then @ol_qty8
        when 8 then @ol_qty9
        when 9 then @ol_qty10
        when 10 then @ol_qty11
        when 11 then @ol_qty12
        when 12 then @ol_qty13
        when 13 then @ol_qty14
        when 14 then @ol_qty15
        else @ol_qty
        end
    ,@s_w_id = case @ol_number
        when 1 then @s_w_id2
        when 2 then @s_w_id3
        when 3 then @s_w_id4
        when 4 then @s_w_id5
        when 5 then @s_w_id6
        when 6 then @s_w_id7
        when 7 then @s_w_id8
        when 8 then @s_w_id9
        when 9 then @s_w_id10
        when 10 then @s_w_id11
        when 11 then @s_w_id12
        when 12 then @s_w_id13
        when 13 then @s_w_id14
        when 14 then @s_w_id15
        else @s_w_id
        end

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

    if (@@rowcount = 0)
       begin
            select @commit_flag = 0
            select NULL, NULL, NULL, NULL
        continue
       end

    /* Otherwise if the item is found */
       update stock
         set s_ytd = s_ytd + @ol_qty,
        @ol_amount = @ol_qty * @i_price,
        @s_quantity = s_quantity - @ol_qty +
              case when (s_quantity - @ol_qty < @ten_smallint)
                then @ninenine_smallint else @zero_smallint
end,
             s_quantity = s_quantity - @ol_qty +
                 case when (s_quantity - @ol_qty <
@ten_smallint)
                 then @ninenine_smallint  else @zero_smallint
end,
         s_order_cnt = s_order_cnt + @one_smallint,
             @s_data = s_data,
             @s_dist = case @local_d_id
                   when 1 then s_dist_01
                   when 2 then s_dist_02
                   when 3 then s_dist_03
                   when 4 then s_dist_04
                   when 5 then s_dist_05
                   when 6 then s_dist_06
                   when 7 then s_dist_07
                   when 8 then s_dist_08
                   when 9 then s_dist_09
                   when 10 then s_dist_10
                   end,
         s_remote_cnt = s_remote_cnt +
         case when (@s_w_id = @w_id)
                    then 0 else 1 end
           where s_w_id = @s_w_id and
                 s_i_id = @i_id

           if (@@rowcount = 0)
             begin
               select @commit_flag = 0
               select NULL, NULL, NULL, NULL
               continue
             end

    INSERT INTO order_line (
        ol_o_id, ol_d_id, ol_w_id, ol_number, ol_i_id,
        ol_supply_w_id, ol_delivery_d, ol_quantity,
        ol_amount, ol_dist_info)
    VALUES (
        @o_id, @d_id, @w_id, @ol_number, @i_id,
        @s_w_id, "18000101", @ol_qty,
        @ol_amount, @s_dist)

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


    SELECT   @c_last  = c_last,
       @c_discount    = c_discount,
       @c_credit  = c_credit,
       @c_ins_id = c_id
    FROM   customer HOLDLOCK
    WHERE   c_w_id  = @w_id
      ANDc_d_id  = @d_id
      ANDc_id   = @c_id

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

    SELECT  @w_tax   = w_tax
    FROM warehouse HOLDLOCK
    WHERE w_id = @w_id

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

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

```
declare @c_last      char(16)

declare @w_street_1 char(20),  @w_street_2 char(20),
   @w_city      char(20),  @w_statechar(2),
   @w_zip   char(9), @w_name      char(10),
   @w_ytd       float,     @w_id_retrieved smallint

declare @d_street_1  char(20),  @d_street_2  char(20),
   @d_city      char(20),  @d_state char(2),
   @d_zip    char(9), @d_name      char(10),
   @d_ytd       float,     @commit_flg  int

declare @c_first   char(16),  @c_middle char(2),
   @c_street_1  char(20),  @c_street_2  char(20),
   @c_city   char(20),  @c_state char(2),
   @c_zip    char(9), @c_phone  char(16),
   @c_sincedatetime, @c_credit   char(2),
   @c_credit_lim   numeric(12,0),   @c_balance   float,
   @c_discount  real,
   @data1    char(250),   @data2    char(250),
   @c_data_1 char(250),   @c_data_2 char(250)

declare @screen_data  char(200),   @today    datetime


BEGIN TRANSACTION PID

   UPDATE district
      SET d_ytd = d_ytd + @h_amount
         ,@d_ytd = d_ytd
         ,@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
         ,@commit_flg = 1
      WHERE d_w_id= @w_id
      AND  d_id   = @d_id

   UPDATE warehouse
      SET w_ytd = w_ytd + @h_amount
         ,@w_ytd = w_ytd
         ,@w_id_retrieved = w_id
         ,@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
      WHERE w_id = @w_id

   if (@@rowcount = 0)
      begin
      select @commit_flg = 0
      end

   /* Customer data */
   UPDATE customer SET
        @c_first = c_first
      , @c_middle = c_middle
      , @c_last = c_last
      , @c_street_1 = c_street_1
      , @c_street_2 = c_street_2
      , @c_city = c_city
      , @c_state = c_state
      , @c_zip = c_zip
      , @c_phone = c_phone
      , @c_credit = c_credit
      , @c_credit_lim = c_credit_lim
      , @c_discount = c_discount
      , c_balance = c_balance - @h_amount
      , @c_balance = c_balance - @h_amount
      , c_ytd_payment = c_ytd_payment + @h_amount
      , c_payment_cnt = c_payment_cnt + 1
      , @c_since = c_since
      , @data1 = c_data1
      , @data2 = c_data2
      , @today = getdate()
   where
      c_id = @c_id
      and c_w_id = @c_w_id
      and c_d_id = @c_d_id

   if (@@rowcount = 0)
      begin
      select @commit_flg = 0
      end

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

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

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

/* COMMIT TRANSACTION PID */

   if (@commit_flg = 1)
           COMMIT TRANSACTION PID
         else
        ROLLBACK TRANSACTION PID

   select    /* Return to client */
      @c_id,
      @c_last,
      @today,
      @w_street_1,
      @w_street_2,
      @w_city,
      @w_state,
      @w_zip,

      @d_street_1,
      @d_street_2,
      @d_city,
      @d_state,
      @d_zip,

      @c_first,
      @c_middle,
      @c_street_1,
      @c_street_2,
      @c_city,
      @c_state,
      @c_zip,
      @c_phone,
      @c_since,
      @c_credit,
      @c_credit_lim,
      @c_discount,
```

```
      @c_balance,
      @screen_data
go
if exists (select * from sysobjects where name =
'payment_byname')
   DROP PROC payment_byname
go
CREATE PROC payment_byname
   @w_id     smallint,  @c_w_id     smallint,
   @h_amount  float,
   @d_id     tinyint,  @c_d_id     tinyint,
   @c_last   char(16)
as
declare @n int,  @c_id   int

declare @w_street_1 char(20),  @w_street_2 char(20),
   @w_city      char(20),  @w_statechar(2),
   @w_zip   char(9), @w_name     char(10),
   @w_ytd       float,     @w_id_retrieved smallint

declare @d_street_1  char(20),  @d_street_2  char(20),
   @d_city      char(20),  @d_state char(2),
   @d_zip    char(9), @d_name      char(10),
   @d_ytd       float,     @commit_flg  int

declare @c_first   char(16),  @c_middle char(2),
   @c_street_1  char(20),  @c_street_2  char(20),
   @c_city   char(20),  @c_state char(2),
   @c_zip    char(9), @c_phone  char(16),
   @c_sincedatetime, @c_credit   char(2),
   @c_credit_lim   numeric(12,0),   @c_balance   float,
   @c_discount  real,
   @data1    char(250),   @data2    char(250),
   @c_data_1 char(250),   @c_data_2 char(250)

declare @screen_data  char(200),   @today    datetime



BEGIN TRANSACTION PNM
   SELECT @n = (count(*)+1)/2
   FROM customer (index c_non1 prefetch $pagesize lru)
HOLDLOCK
   WHERE   c_w_id = @c_w_id and
      c_d_id = @c_d_id and
      c_last = @c_last

   set rowcount @n

   -- @#@ SELECT FROM customer HOLDLOCK
   SELECT @c_id = c_id
   FROM customer (index c_non1 prefetch $pagesize lru)
HOLDLOCK
   WHERE   c_w_id = @c_w_id and
      c_d_id = @c_d_id and
      c_last = @c_last

   -- Reset, so as to do full retrievals hereafter.
   set rowcount 0

   UPDATE district
      SET d_ytd = d_ytd + @h_amount
         ,@d_ytd = d_ytd
         ,@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
         ,@commit_flg = 1
      WHERE d_w_id= @w_id
      AND  d_id   = @d_id

   if (@@rowcount = 0)
      begin
      select @commit_flg = 0
      end

   UPDATE warehouse
      SET w_ytd = w_ytd + @h_amount
         ,@w_ytd = w_ytd
         ,@w_id_retrieved = w_id
         ,@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
      WHERE w_id = @w_id

   /* Customer data */
   UPDATE customer SET
        @c_first = c_first
      , @c_middle = c_middle
      , @c_last = c_last
      , @c_street_1 = c_street_1
      , @c_street_2 = c_street_2
      , @c_city = c_city
      , @c_state = c_state
      , @c_zip = c_zip
      , @c_phone = c_phone
      , @c_credit = c_credit
      , @c_credit_lim = c_credit_lim
      , @c_discount = c_discount
      , c_balance = c_balance - @h_amount
      , @c_balance = c_balance - @h_amount
      , c_ytd_payment = c_ytd_payment + @h_amount
      , c_payment_cnt = c_payment_cnt + 1
      , @c_since = c_since
      , @data1 = c_data1
      , @data2 = c_data2
      , @today = getdate()
   where
      c_id = @c_id
      and c_w_id = @c_w_id
      and c_d_id = @c_d_id

   if (@@rowcount = 0)
      begin
      select @commit_flg = 0
      end

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

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

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

```
/* COMMIT TRANSACTION PNM */

   if (@commit_flg = 1)
           COMMIT TRANSACTION PNM
         else
        ROLLBACK TRANSACTION PNM

   select   /* Return to client */
      @c_id,
      @c_last,
      @today,
      @w_street_1,
      @w_street_2,
      @w_city,
      @w_state,
      @w_zip,

      @d_street_1,
      @d_street_2,
      @d_city,
      @d_state,
      @d_zip,

      @c_first,
      @c_middle,
      @c_street_1,
      @c_street_2,
      @c_city,
      @c_state,
      @c_zip,
      @c_phone,
      @c_since,
      @c_credit,
      @c_credit_lim,
      @c_discount,
      @c_balance,
      @screen_data
go
if exists (select * from sysobjects where name =
'order_status_byid')
   DROP PROC order_status_byid
go
CREATE PROC order_status_byid
   @w_id     smallint,
   @d_id     tinyint,
   @c_id     int
as

DECLARE  @o_id     int,
   @o_entry_d   datetime,
   @o_carrier_idsmallint
```

```
BEGIN TRANSACTION OSID

/* Get the latest order made by the customer */
   set rowcount 1
   SELECT  @o_id = o_id, @o_carrier_id = o_carrier_id,
      @o_entry_d = o_entry_d
   FROM  orders (index o_clu prefetch 16 mru) HOLDLOCK
   WHERE   o_w_id  = @w_id
   AND  o_d_id  = @d_id
   AND  o_c_id  = @c_id
   ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
   set rowcount 0

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

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

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

DECLARE  @o_id     int,
   @o_entry_d   datetime,
   @o_carrier_idsmallint

declare @n int,  @c_id   int
```

```
BEGIN TRANSACTION OSNM
   SELECT @n = (count(*)+1)/2
   FROM customer (index c_non1 prefetch $pagesize lru)
HOLDLOCK
   WHERE   c_w_id = @w_id and
      c_d_id = @d_id and
      c_last = @c_last


   -- Retrieve upto mid-point number of rows.
   set rowcount @n

   -- @#@# SELECT FROM customer HOLDLOCK
   SELECT @c_id = c_id
   FROM customer (index c_non1 prefetch $pagesize lru)
HOLDLOCK
   WHERE   c_w_id = @w_id and
      c_d_id = @d_id and
      c_last = @c_last


/* Get the latest order made by the customer */
   set rowcount 1
   SELECT  @o_id = o_id, @o_carrier_id = o_carrier_id,
      @o_entry_d = o_entry_d
   FROM  orders (index o_clu prefetch 16 mru) HOLDLOCK
   WHERE   o_w_id  = @w_id
   AND  o_d_id  = @d_id
   AND  o_c_id  = @c_id
   ORDER BY o_w_id DESC, o_d_id DESC, o_id DESC
   set rowcount 0

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

   select/* Return single row to client */
      @c_id, c_last, c_first, c_middle, c_balance,
      @o_id,
      @o_entry_d,
      @o_carrier_id
   FROM  customer HOLDLOCK
   WHERE   c_id    = @c_id
```

```
   AND  c_d_id  = @d_id
   AND  c_w_id  = @w_id

COMMIT TRANSACTION OSNM
go

if exists (select * from sysobjects where name = 'delivery')
       drop proc delivery
go

CREATE PROC delivery
       @w_id           smallint,
       @o_carrier_id   smallint,
       @d_id           tinyint = 1
as

declare @no_o_id        int,           @o_c_id         smallint,
       @ol_total        float,         @ol_amount      float,
       @junk_id         smallint, @ten_tinyint   tinyint,
   @one_tinyint tinyint,  @one_smallint  smallint,
       @today           datetime

declare c_del_no CURSOR FOR
       SELECT  no_o_id
       FROM    new_order (index no_clu) HOLDLOCK
       WHERE   no_d_id = @d_id
       AND     no_w_id = @w_id
       FOR UPDATE

       /*
       ** The only purpose of the index hint in the above is to
ensure
       ** that the clustered index is used.  As it turns out, our
optimizer
       ** chooses the clustered index anyway -- with or without
the hint.
       */
begin
   SELECT @one_tinyint = 1, @ten_tinyint = 10,
@one_smallint = 1
   while (@d_id <= @ten_tinyint ) begin
   BEGIN TRANSACTION DEL

       OPEN c_del_no

       FETCH c_del_no INTO @no_o_id

       if (@@sqlstatus != 0)
       begin
              COMMIT TRANSACTION DEL
              select NULL
              CLOSE c_del_no
       end
       else
       begin
              DELETE  FROM new_order
              WHERE   CURRENT OF c_del_no
              CLOSE   c_del_no

              SELECT  @ol_total = 0.0, @today = getdate()

       -- @#@# UPDATE order_line
              UPDATE order_line
              SET       ol_delivery_d = @today
                     , @ol_total = @ol_total + ol_amount
              WHERE   ol_o_id = @no_o_id
              AND     ol_d_id = @d_id
       AND  ol_w_id = @w_id

       -- @#@# UPDATE orders
              UPDATE  orders
              SET       o_carrier_id = @o_carrier_id
                     , @o_c_id = o_c_id
              WHERE   o_id    = @no_o_id
              AND     o_d_id  = @d_id
              AND     o_w_id  = @w_id

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

              COMMIT TRANSACTION DEL

              select      /* Return to client */
                     @no_o_id
       end
   SELECT @d_id = @d_id + @one_tinyint
   end
end
go


if exists ( SELECT name FROM sysobjects WHERE name =
'stock_level')
   DROP PROC stock_level
go

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

if [ "$?" != "0" ]
then
   echo ""
   echo " **** WARNING: Could not connect to server to
install procs! ****"
   echo " ****        SQL SErver must have failed!!! ****"
   echo " ****        TPC-C build & run will fail!!! ****"
   echo ""
fi
```

tpcc_unpartition.sh

```
#!/bin/sh -f

isql -Usa -P$PASSWORD << EOF &
use tpcc
go
declare @dbid   int
select @dbid = db_id("tpcc")
alter table new_order unpartition
--dbcc tune("notforload", @dbid, new_order)
go
EOF

isql -Usa -P$PASSWORD << EOF &
use tpcc
go
declare @dbid   int
select @dbid = db_id("tpcc")
alter table orders unpartition
--dbcc tune("notforload", @dbid, orders)
go
EOF
```

```
isql -Usa -P$PASSWORD << EOF &
use tpcc
go
declare @dbid   int
select @dbid = db_id("tpcc")
alter table order_line unpartition
--dbcc tune("notforload", @dbid, order_line)
go
EOF

wait
```

start_bld_system.sh

```
#!/bin/bash

. ./bm_setup
export BLD_SYSTEM=1
export verstr=$version

echo $verstr

./run_server - -c./load_15_0_build.cfg -T3474
```

install_sp_help.sh

```
#!/bin/sh -x

isql -Usa -P -e <<EOF

dbcc traceon(7730)
go

use sybsystemprocs
go

exec sp_drop_object 'sp_helpartition', 'procedure'
go

if exists (select *
      from sysobjects
            where sysstat & 7 = 4
                  and name = 'sp_helpartition')
begin
      drop procedure sp_helpartition
end
EOF
isql -Usa -P -e <<EOF
use sybsystemprocs
go

create procedure sp_helpartition
@tabname varchar(767) = NULL,
@indexname varchar(255) = NULL,
@partitionname varchar(255) = NULL
as

declare @tableid  int,
   @indexid int,
   @dbid     int,
   @use_savedname  int,
   @columnname  varchar(255),
   @columnlist   varchar(1024),
   @indid     smallint,
   @current_indid smallint,
   @indextype varchar(16),
   @datapage_indid smallint,
   @ptntype smallint,
   @typename   varchar(12),
   @condid     int,
   @range    smallint,
   @hash     smallint,
   @roundrobin  smallint,
   @list     smallint,
   @config_parm   int,
   @hiddenmsg varchar(255),
   @disallowmsg   varchar(255),
   @ptnid     int,
   @avg_pages int,
   @max_pages int,
   @min_pages int,
   @datapage_ptnid int

if @@trancount = 0
begin
   set transaction isolation level 1
   set chained off
end


select @range = v.number from master.dbo.spt_values v
      where v.type = 'PN' and v.name = 'range'
select @hash = v.number from master.dbo.spt_values v
      where v.type = 'PN' and v.name = 'hash'
select @roundrobin = v.number from master.dbo.spt_values
v
      where v.type = 'PN' and v.name = 'roundrobin'
select @list = v.number from master.dbo.spt_values v
      where v.type = 'PN' and v.name = 'list'

set nocount on

if @tabname is null
begin

select "owner" = user_name(uid), o.name
   , "partitions" = (select count(*)
         from syspartitions p
         where p.id = o.id
           and p.indid < 2 )
   , "partition_type" = (select v.name
            from master.dbo.spt_values v
            where v.type = 'PN'
          and v.number = isnull(i.partitiontype, @roundrobin))
into #all_tables
from sysobjects o, sysindexes i
where o.type = "U" and o.id = i.id and i.indid < 2

exec sp_autoformat @fulltabname = "#all_tables", @orderby
= "order by 1, 2"
-- order by 1,2

return 0

end

if @tabname like '%.%.%' and
   substring(@tabname, 1, charindex('.', @tabname) - 1) !=
db_name()
begin
        /* 17460, "Object must be in the current database." */
        raiserror 17460
        return (1)
end

if not exists (select 1 from sysobjects
         where id = object_id(@tabname)
           and (type = 'S' or type = 'U'))
begin
   /** 17733, "There is no table named '%1!'."*/
   raiserror 17733, @tabname
   return (1)
end

select @tableid = object_id(@tabname)
   , @dbid = db_id()




if @indexname is not null and @indexname != 'all' and
   not exists (select 1
         from sysindexes
               where id = @tableid
                 and name = @indexname)
```

```
begin
        /* 17734, "There is no index named '%1!' for table
'%2!'."*/
        raiserror 17734, @indexname, @tabname
        return (1)
end

if @indexname != 'all'
begin
   select @indexid = indid
      from sysindexes
      where id = @tableid
           and ((@indexname is null and indid < 2)
           or name = @indexname)

   /*
   ** If user didn't specify index name, it implicitly
   ** means the base table, but for table with clustered
   ** index, the partition name for base table is moved
   ** to savedname in syspartitions. In this case, we need
   ** to check the savedname to see if the partition name
   ** given is valid.
   */
   if @indexname is null and @indexid = 1
      select @use_savedname = 1
   else
      select @use_savedname = 0

   if @partitionname is not null and
      not exists (select 1
              from syspartitions
                    where id = @tableid
              and indid = @indexid
              and ((@use_savedname = 0
              and name = @partitionname)
              or (@use_savedname = 1
                 and cdataptnname = @partitionname)))
   /*
   **  Partition doesn't exist on this index so return.
   */
   begin
      /*
      ** 19305, "There is no partition named '%1!' for table
'%2!',
      ** index '%3!'."
      */
      if @indexname is null
         raiserror 19305, @partitionname, @tabname,
@tabname
      else
         raiserror 19305, @partitionname, @tabname,
@indexname
      return (1)
   end
end

declare keycursor cursor for
select distinct c.name, k.indid
from syspartitionkeys k, syscolumns c
where k.id = @tableid
      and k.id = c.id
      and c.colid = k.colid
order by k.indid, k.position
for read only

create table #col_names (
   indid smallint not null,
   col_name_list varchar(1024) null)
lock allpages


create unique clustered index col_names_uniqind on
#col_names(indid)

open keycursor


select @columnlist = "", @current_indid = -1

fetch keycursor into @columnname, @indid

while(@@sqlstatus = 0)
begin
   /* If first row, set current_indid and columnlist */
   if (@@rowcount = 1)
   begin
          select @columnlist = @columnname
          select @current_indid = @indid
   end
   /* If next row of same index, concate the columnlist */
       else if (@current_indid = @indid)
       begin
             select @columnlist = @columnlist + ', ' +
@columnname
       end

   /*
   ** If next index, insert the columnlist for previous index,
   ** and reset the current_indid and columnlist.
   */
       else
       begin
             insert #col_names values
                    (@current_indid, @columnlist)
             select @columnlist = @columnname
             select @current_indid = @indid
       end

   fetch keycursor into @columnname, @indid

   /*
   ** If end of cursor, just insert the columnlist for previous
   ** index, if there had been one.
   */
   if (@@sqlstatus = 2)
   begin
      insert #col_names values
            (@current_indid, @columnlist)
   end
end

close keycursor

deallocate cursor keycursor
update #col_names set col_name_list = NULL
   where exists (select 1 from sysindexes i,
master.dbo.spt_values v
           where     i.id = @tableid
            and i.indid = #col_names.indid
            and isnull(i.partitiontype, @roundrobin) = v.number
            and v.type = 'PN'
            and v.name = 'roundrobin'
          )

insert #col_names select i.indid, NULL
       from sysindexes i
       where id = @tableid and not exists
        (select 1
        from syspartitionkeys k
        where k.id = @tableid and i.indid = k.indid)

if (@indexname = 'all')
begin
   select 'name' = i.name,
      'type' = case when i.indid = 0 then 'base table'
             when i.indid = 255 then 'text/image'
             when i.status3 & 8 = 8 then 'local index'
             else 'global index'
         end,
      'partition_type' = v.name,
      'partitions' = (select count(*)
           from syspartitions p
           where p.id = @tableid
             and p.indid = i.indid

             ),
```

```
      'partition_keys' = n.col_name_list
   into #result_all lock allpages
   from sysindexes i, #col_names n, master.dbo.spt_values v
   where i.id = @tableid
     and i.indid = n.indid
     and v.number = isnull(i.partitiontype, @roundrobin)
     and v.type = 'PN'

   exec sp_autoformat #result_all

   return (0)
end

select @ptntype = isnull(i.partitiontype, @roundrobin),
       @condid = i.conditionid,
       @columnlist = n.col_name_list,
       @indextype =
       case
         when i.indid = 0 or @indexname is null
            then 'base table'
         when i.indid = 255 then 'text/image'
         when i.status3 & 8 = 8 then 'local index'
         else 'global index'
       end
from sysindexes i, #col_names n
where i.id = @tableid
   and i.indid = @indexid
   and i.indid = n.indid

if (@indexname is null)
   select @datapage_indid = 0
else
   select @datapage_indid = @indexid

select @typename = v.name
from master.dbo.spt_values v
where v.type = 'PN' and v.number = @ptntype

select @hiddenmsg = NULL
select @disallowmsg = NULL

if @ptntype in (@range, @list)
begin
   /*
   ** If the configuration parameter 'allow select on
syscomments.text'
   ** is set to 0, then the user can access the text ONLY in the
   ** following cases
   **
   **      1. if the user has sa_role
   **      2. if the object is owned by the user
   **
   */
   select @config_parm = value
      from master.dbo.syscurconfigs
      where config = 258

   if @config_parm = 0 and user_id() != 1
      and not exists (select name from sysobjects
               where uid = user_id()
               and id = @tableid)
   begin
      exec sp_getmessage 19340, @disallowmsg output
   end

   /* See if the partition condition text is hidden */
   else if exists (select 1 from syscomments c
         where c.id = @condid and (c.status & 1 = 1))
   begin
      exec sp_getmessage 19337, @hiddenmsg output
   end
end

select 'name' = case
         when @indexname is not null then @indexname
         else @tabname
       end,
   'type' = @indextype,
   'partition_type' = @typename,
   'partitions' = (select count(*) from syspartitions p
         where p.id = @tableid
           and p.indid = @indexid),
   'partition_keys' = @columnlist
into #result_head lock allpages

exec sp_autoformat #result_head
print " "


create table #result_body(partition_name varchar(255) not
null,
          partition_id   int not null,
          pages int not null,
          row_count numeric(18,0) not null,
          segment varchar(255) not null,
          create_date datetime not null,
          seqnum int not null)
          lock allpages

create table #result_cond(Partition_Conditions varchar(255)
null,
          seqnum int not null,
          colid int not null,
          colid2 int not null)
          lock allpages

if (@partitionname is not null)
begin

   select @ptnid = partitionid
   from syspartitions
   where id = @tableid
         and indid = @indexid
           and ((@use_savedname = 1 and cdataptnname =
@partitionname)
         or name = @partitionname)
   insert #result_body
      select  'partition_name' =
         case
           when @use_savedname = 1 then p.cdataptnname
           else p.name
         end,
         @ptnid,
         data_pages(@dbid, @tableid, @datapage_indid,
@ptnid),
         case
           when (@datapage_indid > 1) then 0
           else row_count(@dbid, @tableid, @ptnid)
         end,
         s.name,
         p.crdate,
         0
      from syspartitions p, syssegments s
      where p.id = @tableid
         and p.indid = @indexid
         and p.segment = s.segment
         and p.partitionid = @ptnid

   /* Get partition condition */
   if @ptntype in (@roundrobin, @hash)
      insert #result_cond values (NULL,0,0,0)
   else if (@disallowmsg is null) and (@hiddenmsg is null)
      insert #result_cond
         select c.text, c.number, c.colid, c.colid2
         from syscomments c
         where c.id = @condid
               and c.partitionid = @ptnid

end

else
begin
   if (@ptntype = @roundrobin)
   begin
      insert #result_body
      select  'partition_name' =
```

```
      case
        when @use_savedname = 1
          then p.cdataptnname
        else p.name
      end,
      p.partitionid,
      data_pages(@dbid, @tableid, @datapage_indid,
          p.partitionid),
      case
        when (@datapage_indid > 1) then 0
        else row_count(@dbid, @tableid, p.partitionid)
      end,
      (select s.name from syssegments s
       where s.segment = p.segment),

          p.crdate,

      0
    from syspartitions p
    where p.id = @tableid
            and p.indid = @indexid
end
else
begin
    insert #result_body
    select 'partition_name' =
      case
        when @use_savedname = 1
          then p.cdataptnname
        else p.name
      end,
      p.partitionid,
      data_pages(@dbid, @tableid, @datapage_indid,
          p.partitionid),
      case
        when (@datapage_indid > 1) then 0
        else row_count(@dbid, @tableid, p.partitionid)
      end,
      s.name,

      p.crdate,

      (select distinct c.number
       from syscomments c
       where c.id = @condid
         and c.partitionid = p.partitionid)

    from syspartitions p, syssegments s
    where p.id = @tableid
          and p.indid = @indexid
         and p.segment = s.segment
end
```

```
  /* Get partition condition */
  if (@ptntype = @roundrobin or @ptntype = @hash)
     insert #result_cond values (NULL,0,0,0)
  else if (@disallowmsg is null) and (@hiddenmsg is null)
     insert #result_cond
     select c.text, c.number, c.colid, c.colid2
     from syscomments c
     where c.id = @condid
end

if (@indexid > 1)
begin --{
  if (@indextype = 'text/image')
  begin --{
       /*
       ** This should be changed when we provide
       ** per partition text partition to display
       ** the number of rows in the corresponding
       ** data partition.
       */
                update #result_body
                set row_count = row_count(@dbid, @tableid)
  end --}
  else if (@indextype = 'global index')
     begin --{
          update #result_body
          set row_count
               = row_count(@dbid, @tableid)
     end --}
     else
     begin --{
          if (@partitionname is not null)
          begin
               select @datapage_ptnid = data_partitionid
               from syspartitions
               where id = @tableid and
                     partitionid = @ptnid
               update #result_body
               set row_count = row_count(@dbid, @tableid,
               @datapage_ptnid)
          end
          else
          begin
               update #result_body
               set row_count = row_count(@dbid, @tableid,
               s.data_partitionid)
               from syspartitions as s, #result_body as r
               where s.id = @tableid and
                     s.partitionid = r.partition_id
          end
     end --}
```

```
end --}

exec sp_autoformat #result_body, 'partition_name,
partition_id, pages,row_count,segment,create_date', NULL,
'order by seqnum, partition_id'

print " "

if @disallowmsg is not null
  print @disallowmsg
else
begin
  if @hiddenmsg is not null
    insert #result_cond values (@hiddenmsg,0,0,0)

  exec sp_autoformat #result_cond, 'Partition_Conditions',
    NULL, 'order by seqnum,colid2,colid'
end

if @partitionname is not null
  return (0)

print " "

select @avg_pages = avg(data_pages(@dbid, @tableid,
@datapage_indid,
          partitionid)),
     @max_pages = max(data_pages(@dbid, @tableid,
@datapage_indid,
          partitionid)),
     @min_pages = min(data_pages(@dbid, @tableid,
@datapage_indid,
          partitionid))
from syspartitions
where id = @tableid
  and indid = @indexid

select 'Avg_pages' = @avg_pages,
     'Max_pages' = @max_pages,
     'Min_pages' = @min_pages,
     'Ratio(Max/Avg)' =
     case
       when @avg_pages = 0 then 0
       else convert(float, @max_pages)/@avg_pages
     end,
  'Ratio(Min/Avg)' =
     case
       when @avg_pages = 0 then 0
       else convert(float, @min_pages)/@avg_pages
     end

set nocount off
```

```
return (0)
go
exec sp_procxmode 'sp_helpartition', 'anymode'
go
grant execute on sp_helpartition to public
go
dump tran master with truncate_only
go
dump transaction sybsystemprocs with truncate_only
go
EOF
isql -Usa -P<<EOF
use tpcc
go
sp_helpartition item
go
dbcc traceoff(7730)
go

EOF
```

resize_logs.sh

```
#!/bin/sh

isql -Usa -P -e <<EOF

sp_helpdevice 'logdisk2'
go

disk resize name = 'logdisk2', size = 25088000
go

alter database tpcc log on logdisk2 = 49000
go

sp_helpdevice 'logdisk3'
go

disk resize name = 'logdisk3', size = 25088000
go

alter database tpcc log on logdisk3 = 49000
go

EOF
```

history.500.sh

```
#!/bin/sh

isql -Usa -P -e<<EOM
use tpcc
go
if exists ( select name from sysobjects where name =
'history' )
        drop table history
go
create table history (
        h_c_id          int,
        h_c_d_id        tinyint,
        h_c_w_id        smallint,
        h_d_id          tinyint,
        h_w_id          smallint,
        h_date          datetime,
        h_amount        float,
        h_data          char(24)
) on Shistory
go
alter table history partition 500
go
EOM
/tpcc/products/tpcc11/loader/load.15_0 history 1 22000 "" 500
```

12.2. Database Load

tpcc_ld_loader.h

```
#ifndef TPCC_INCLUDED
#define TPCC_INCLUDED

#include "sybfront.h"
#include "sybdb.h"
#ifdef _NTINTEL
#include <sys/timeb.h>
#include <stdlib.h>
#include <process.h>
#endif
#include <time.h>


/* Population constants */
#ifdef CACHED
#define MAXITEMS 10000
#define CUST_PER_DIST 300
#define DIST_PER_WARE 10
#define ORD_PER_DIST 300
#else
#define MAXITEMS 100000
#define CUST_PER_DIST 3000
#define DIST_PER_WARE 10
#define ORD_PER_DIST 3000
#endif

#define NURAND_C 123

/* Types of application variables */
typedef int     COUNT;
typedef int     ID;
typedef double  MONEY;
typedef double  FLOAT;
typedef char    TEXT;
typedef struct { int x[2];} DATE;
typedef int     LOGICAL;

typedef enum
   {COUNT_T, ID_T, MONEY_T, FLOAT_T, TEXT_T,
    DATE_T, LOGICAL_T, MAX_T}
   DATA_TYPE;

typedef struct timeval TIME;


#define YES 1
#define NO 0
#define EOF (-1)

#ifndef NULL
#define NULL ((void *)0)
#endif

#define INVALID_SLICE_NUM  -1

#ifdef DEBUG
#define debug printf
#else
#define debug donothing
#endif

/* define function types */

CS_INT  cl_err_handler(CS_CONTEXT      *context,
CS_CONNECTION   *connection, CS_CLIENTMSG
*errmsg);
CS_INT server_msg_handler( CS_CONTEXT     *context,
CS_CONNECTION  *connection, CS_SERVERMSG
*srvmsg);
CS_INT cs_err_handler( CS_CONTEXT      *context,
CS_CLIENTMSG    *errmsg);
extern   int   batch_size;

#endif /* TPCC_INCLUDED */

#ifdef _NTINTEL
```

```
#define drand48() rand()
#endif
void LoadWarehouse() ;
void begin_warehouse_load() ;
void warehouse_load() ;
void end_warehouse_load() ;
void LoadDistrict() ;
void begin_district_load() ;
void district_load() ;
void end_district_load() ;
void LoadItems() ;
void begin_item_load() ;
void item_load() ;
void end_item_load() ;
void LoadHist() ;
void LoadCustHist() ;
void begin_history_load() ;
void history_load() ;
void end_history_load() ;
void LoadCustomer() ;
void Customer() ;
void begin_customer_load() ;
void customer_load() ;
void end_customer_load() ;
void LoadOrd() ;
void LoadNew() ;
void Orders() ;
void OrderLine() ;
void NewOrder() ;
void begin_order_load() ;
void order_load() ;
void end_order_load() ;
void begin_order_line_load() ;
void order_line_load() ;
void end_order_line_load() ;
void begin_new_order_load() ;
void new_order_load() ;
void end_new_order_load() ;
void LoadStock() ;
void begin_stock_load() ;
void stock_load() ;
void end_stock_load() ;
void test() ;
void getargs() ;
void MakeAddress() ;
void LastName() ;
void Original() ;
void RandomPermutation() ;
void Randomize() ;
void datetime();
void bulk_null() ;
void bulk_non_null() ;
void bulk_close() ;
int MakeNumberString() ;
int MakezipString() ;
int MakeAlphaString() ;
int RandomNumber() ;
int NURandomNumber() ;
int bulk_open() ;
CS_RETCODE bulk_load() ;
CS_RETCODE bulk_bind (int, int, char *, void *, int,
CS_BLKDESC *);
void donothing(const char *, ...);
```

tpcc_ld_load.c

```
typedef unsigned longBitVector;
#define WSZ     (sizeof(BitVector)*8)
/* For axposf use WAREBATCH of 144 */
#ifndef WAREBATCH
#ifdef _NTINTEL
#define WAREBATCH   288
#else
#define WAREBATCH   10000
#endif
#endif
#define nthbit(map,n) map[(n)/WSZ] &  (((BitVector)0x1)<<
((n)%WSZ))
#define setbit(map,n) map[(n)/WSZ] |= (((BitVector)0x1)<<
((n)%WSZ))

/*************************************************************
Load TPCC tables
*************************************************************/
#include "stdio.h"
#include <stdlib.h>
#include <ctpublic.h>
#include <bkpublic.h>
#include <unistd.h>
#include "string.h"
#include "tpcc_ld_loader.h"

int load_item;
int load_warehouse;
int load_district;
int load_history;
int load_orders;
int load_new_order;
int load_order_line;
int load_customer;
int load_stock;
ID  w1, w2;
ID  i1, i2;
ID  warehouse;
int batch_size = 1000;
char   password[10];
int  sliceNum = INVALID_SLICE_NUM;
int  numOfSlices = INVALID_SLICE_NUM;
int  loadSlices = 1;
int  perm[MAXITEMS+1];
CS_BLKDESC    *blkdesc, *blkdesc1;

#ifdef _NTINTEL
   BitVector * bmp;
#else
   BitVector original[WAREBATCH][((MAXITEMS+(WSZ-
1))/WSZ)], * bmp;
#endif


#define print_prologue(name)    \
   if(1)\
   {\
      printf("Loading %s table from warehouse %d to %d into
partition %d\n", (name),  w1, w2, sliceNum);\
   }\
   else
#define print_prologue_item(name)     \
   if(1)\
   {\
      printf("Loading %s table from warehouse %d to %d and
from item %d to %d into partition %d\n", (name),  w1, w2, i1,
i2, sliceNum);\
   }\
   else

int main(argn, argv)
   int argn;
   char **argv;
{
   ID starting_wid, ending_wid, winc, last_wid;
   ID starting_iid, ending_iid, iinc;
   int current_slicenum;
   pid_t pid = 0, parallel_parent = 0;



   getargs(argn, argv);
   /* This is needed globally,  because only 10% of the total
item
   ** must be marked Original and we need to create a global
   ** permutation so that all the threads look into the same
   ** permutation array.
   */
```

```
  if (load_item)
  {
     RandomPermutation(perm, MAXITEMS);
  }

  /*
  ** Except for item and stock all others are parallely
  ** loaded based on the warehouse id. Item and Stock
  ** are loaded parallely based on the item.
  **
  ** We make sure that at least 2 waehouses are given
  ** to each thread. For Item and Stock we don't need
  ** to do this check, because we will be sharing the
  ** load of MAXITEMS equally among all threads.
  */
  if ( numOfSlices != INVALID_SLICE_NUM )
  {
     if (!load_item
     && !load_stock
     && (w2 < (numOfSlices*2)))
     {
        printf(" Please use at most %d slices for loading %d
warehouses. Aborting load for table %s.\n", w2/2, w2,
argv[1]);
        exit(1);
     }

     winc = w2 / numOfSlices;
     iinc = MAXITEMS / numOfSlices;

     /*
     ** One restriction that we are posing is that, number of
     ** warehouses  should be divisible by
     ** number of slices. This is because some bug reported
     ** by Sun while doing a parallel loading with 5000
warehouses
     ** and slicing across 6 partitions where each slice gets
     ** 833.3. To get around this problem we won't allow
fractions.
     */

     if( !( load_item || load_stock) && ((w2 % numOfSlices) !=
0) )
     {
        printf(" Please use the number of slices which divides
%d evenly.\n", w2 );
        exit(1);
     }

     last_wid = w2;
     for( starting_wid = w1, ending_wid = winc,
current_slicenum =1,
        starting_iid = 1, ending_iid = iinc;
        current_slicenum <= numOfSlices;
        starting_wid += winc,
        ending_wid += winc,
        starting_iid += iinc,
        ending_iid += iinc,
        current_slicenum++)
     {
        if ( current_slicenum == numOfSlices )
        {
           /* Last thread. Give the remaining */
           ending_wid = last_wid;
           ending_iid = MAXITEMS;
        }
        if( loadSlices )
        {
           sliceNum = current_slicenum;
        }
        if ( load_item || load_stock)
        {
           i1 = starting_iid;
           i2 = ending_iid;
           printf("Loading from item %d to %d on Slice num %d
\n", i1, i2, sliceNum);
        }
        else
        {
           w1 = starting_wid;
           w2 = ending_wid;
           printf("Loading from warehouse %d to %d on Slice
num %d \n", w1, w2, sliceNum);
        }
        if ( (pid = fork()) < 0 )
        {
           printf("Fork failed \n");
           exit(1);
        }else if ( pid > 0 )
        {
           parallel_parent = 1;
           /* Parent */
           continue;
        }else
        {
           parallel_parent = 0;
           /* child */
           break;
        }
     }
  }

  if ( parallel_parent )
  {
        while( wait(0) != -1 );
        printf("Done Loading \n");
        exit(0);
  }
  else
  {
     Randomize();
     if (load_item)     LoadItems();
     if (load_warehouse) LoadWarehouse(w1, w2);
     if (load_district)  LoadDistrict(w1, w2);
     if (load_history)  LoadHist(w1, w2);
     if (load_customer) LoadCustomer(w1, w2);
     if (load_stock)    LoadStock(w1, w2);
     if (load_orders)   LoadOrd(w1, w2);
     if (load_new_order)  LoadNew(w1, w2);
     return 0;
  }
}


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

Warehouse

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


ID   w_id;
TEXT w_name[10+1];
TEXT w_street_1[20+1];
TEXT w_street_2[20+1];
TEXT w_city[20+1];
TEXT w_state[2+1];
TEXT w_zip[9+1];
FLOAT w_tax;
MONEY w_ytd;


int bulk_w;
void
LoadWarehouse(w1, w2)
   ID w1, w2;
{

   print_prologue("Warehouse");
   bulk_w = bulk_open("tpcc", "warehouse", password,
&blkdesc, &sliceNum);
```

```
   for (warehouse=w1; warehouse<=w2; warehouse++)
   {
      printf("Loading warehouse for warehouse %d\n",
warehouse);

      w_id = warehouse;
      MakeAlphaString(6, 10, w_name);
      MakeAddress(w_street_1, w_street_2, w_city, w_state,
w_zip);

      w_tax = RandomNumber(0, 2000) / 10000.0;
      w_ytd = 300000.00 * 100;

      begin_warehouse_load();
      warehouse_load();

      printf("loaded warehouse for warehouse %d\n",
warehouse);
   }
   end_warehouse_load();
   return;
}

void
begin_warehouse_load()
{
   int   i = 1;


   bulk_bind(bulk_w, i++, "w_id",   &w_id, ID_T,blkdesc);
   bulk_bind(bulk_w, i++, "w_name", w_name,
TEXT_T,blkdesc);
   bulk_bind(bulk_w, i++, "w_street_1", w_street_1,
TEXT_T,blkdesc);
   bulk_bind(bulk_w, i++, "w_street_2", w_street_2,
TEXT_T,blkdesc);
   bulk_bind(bulk_w, i++, "w_city", w_city, TEXT_T,blkdesc);
   bulk_bind(bulk_w, i++, "w_state", w_state,
TEXT_T,blkdesc);
   bulk_bind(bulk_w, i++, "w_zip", w_zip, TEXT_T,blkdesc);
  bulk_bind(bulk_w, i++, "w_tax", &w_tax, FLOAT_T,blkdesc);
   bulk_bind(bulk_w, i++, "w_ytd", &w_ytd,
MONEY_T,blkdesc);
   return;
}

void
warehouse_load()
{
   debug("Loading Warehouse %d\n", w_id);
   bulk_load(bulk_w, blkdesc);
}

void
end_warehouse_load()
{
   bulk_close(bulk_w, blkdesc);
}



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

District

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



ID d_id;
ID d_w_id;
TEXT d_name[10+1];
TEXT d_street_1[20+1];
TEXT d_street_2[20+1];
TEXT d_city[20+1];
TEXT d_state[2+1];
TEXT d_zip[9+1];
FLOAT d_tax;
MONEY d_ytd;
ID   d_next_o_id;

int bulk_d;

void
LoadDistrict(w1, w2)
   ID w1, w2;
{
   ID w_id;

   print_prologue("District");
   bulk_d = bulk_open("tpcc", "district", password, &blkdesc,
&sliceNum);
   for (w_id=w1; w_id<=w2; w_id++)
   {
      printf("Loading districts for warehouse %d\n", w_id);

      d_w_id = w_id;
      d_ytd = 30000.00 * 100;
      d_next_o_id = 3001;

      for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
      {

         MakeAlphaString(6, 10, d_name);
         MakeAddress(d_street_1, d_street_2, d_city, d_state,
d_zip);
         d_tax = RandomNumber(0,2000) / 10000.0;

         begin_district_load();
         district_load();
      }
      printf("loaded district for warehouse %d\n", w_id);
   }
   end_district_load();
   return;
}



void
begin_district_load()
{
   int   i = 1;


   bulk_bind(bulk_d, i++, "d_id", &d_id, ID_T,blkdesc);
   bulk_bind(bulk_d, i++, "d_w_id", &d_w_id, ID_T,blkdesc);
   bulk_bind(bulk_d, i++, "d_name", d_name,
TEXT_T,blkdesc);
   bulk_bind(bulk_d, i++, "d_street_1", d_street_1,
TEXT_T,blkdesc);
   bulk_bind(bulk_d, i++, "d_street_2", d_street_2,
TEXT_T,blkdesc);
   bulk_bind(bulk_d, i++, "d_city", d_city, TEXT_T,blkdesc);
  bulk_bind(bulk_d, i++, "d_state", d_state, TEXT_T,blkdesc);
   bulk_bind(bulk_d, i++, "d_zip", d_zip, TEXT_T,blkdesc);
   bulk_bind(bulk_d, i++, "d_tax", &d_tax, FLOAT_T,blkdesc);
  bulk_bind(bulk_d, i++, "d_ytd", &d_ytd, MONEY_T,blkdesc);
   bulk_bind(bulk_d, i++, "d_next_o_id", &d_next_o_id,
ID_T,blkdesc);
   return;
}
void
district_load()
{
   debug("District d_id=%d  w_id=%d d_name=%s
d_street_1=%s d_street_2=%s d_city=%s d_state=%s
d_zip=%s d_tax=%f d_ytd=%f d_next_o_id=%d\n", d_id,
```

```
d_w_id, d_name, d_street_1, d_street_2, d_city, d_state,
d_zip, d_tax, d_ytd, d_next_o_id);
   bulk_load(bulk_d, blkdesc);
   return;
}
void
end_district_load()
{
   bulk_close(bulk_d, blkdesc);
   return;
}



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

Item

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



ID i_id;
ID i_im_id;
TEXT i_name[24+1];
MONEY i_price;
TEXT  i_data[50+1];

int bulk_i;
void
LoadItems()
{

   print_prologue_item("Item");

   bulk_i = bulk_open("tpcc", "item", password, &blkdesc,
&sliceNum);

   /* select exactly 10% of items to be labeled "original" */

   /* do for each item */
   for (i_id=i1;   i_id <= i2; i_id++)
   {

      /* Generate Item Data */
      MakeAlphaString(14, 24, i_name);
      i_price = RandomNumber(100,10000);
      MakeAlphaString(26, 50, i_data);
      if (perm[i_id] <= (MAXITEMS+9)/10)
         Original(i_data);

      /* Generate i_im_id for V 3.0 */
      i_im_id = RandomNumber(1, 10000);

      begin_item_load();
      item_load();
    }

   end_item_load();
   return;
}


void
begin_item_load()
{
   int   i = 1;


   /* bind the variables to the sybase columns */
   bulk_bind(bulk_i, i++, "i_id", &i_id, ID_T,blkdesc);
   bulk_bind(bulk_i, i++, "i_im_id", &i_im_id, ID_T,blkdesc);
   bulk_bind(bulk_i, i++, "i_name", i_name, TEXT_T,blkdesc);
   bulk_bind(bulk_i, i++, "i_price", &i_price,
MONEY_T,blkdesc);
   bulk_bind(bulk_i, i++, "i_data", i_data, TEXT_T,blkdesc);
   return;
}



void
item_load()
{
   debug("i_id=%3d price=%5.2f data=%s\n",
      i_id, i_price, i_data);
   bulk_load(bulk_i, blkdesc);
   return;
}

void
end_item_load()
{
   bulk_close(bulk_i, blkdesc);
   return;
}


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

History

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



ID h_c_id;
ID h_c_d_id;
ID h_c_w_id;
ID h_d_id;
ID h_w_id;
DATE h_date;
MONEY h_amount;
TEXT h_data[24+1];

int bulk_h;
void
LoadHist(w1, w2)
   ID w1, w2;
{
   ID w_id;
   ID d_id, c_id;

   print_prologue("History");
   bulk_h = bulk_open("tpcc", "history", password, &blkdesc,
&sliceNum);
   for (w_id=w1; w_id<=w2; w_id++)
   {

      for (d_id=1;  d_id <= DIST_PER_WARE; d_id++)
      {
         for (c_id=1; c_id <= CUST_PER_DIST; c_id++)
            LoadCustHist(w_id, d_id, c_id);
      }

      printf("\nLoaded history for warehouse %d\n", w_id);
   }
   end_history_load();
   return;
}

void
LoadCustHist(w_id, d_id, c_id)
   ID w_id, d_id, c_id;
{

   h_c_id = c_id;
   h_c_d_id = d_id;
   h_c_w_id = w_id;
```

```
   h_d_id = d_id;
   h_w_id = w_id;
   h_amount = 10.0 * 100;
   MakeAlphaString(12, 24, h_data);
   datetime(&h_date);
   begin_history_load();
   history_load();
   return;
}


void
begin_history_load()
{
   int  i = 1;


   bulk_bind(bulk_h, i++, "h_c_id",   &h_c_id,   ID_T,blkdesc);
   bulk_bind(bulk_h, i++, "h_c_d_id", &h_c_d_id,
ID_T,blkdesc);
   bulk_bind(bulk_h, i++, "h_c_w_id", &h_c_w_id,
ID_T,blkdesc);
   bulk_bind(bulk_h, i++, "h_d_id",   &h_d_id,   ID_T,blkdesc);
  bulk_bind(bulk_h, i++, "h_w_id",   &h_w_id,   ID_T,blkdesc);
   bulk_bind(bulk_h, i++, "h_date",   &h_date,
DATE_T,blkdesc);
   bulk_bind(bulk_h, i++, "h_amount", &h_amount,
MONEY_T,blkdesc);
   bulk_bind(bulk_h, i++, "h_data",   h_data,
TEXT_T,blkdesc);
   return;
}
void
history_load()
{

   debug("h_c_id=%d  h_amount=%g\n", h_c_id, h_amount);
   bulk_load(bulk_h, blkdesc);
   return;
}

void
end_history_load()
{
   bulk_close(bulk_h, blkdesc);
   return;
}
```

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

Customer

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

/* static variables containing fields for customer record */
ID c_id;
ID c_d_id;
ID c_w_id;
TEXT c_first[16+1];
TEXT c_middle[2+1] = "OE";
TEXT c_last[16+1];
TEXT c_street_1[20+1];
TEXT c_street_2[20+1];
TEXT c_city[20+1];
TEXT c_state[2+1];
TEXT c_zip[9+1];
TEXT c_phone[16+1];
DATE c_since;
TEXT c_credit[2+1] = "?C";
MONEY c_credit_lim = 50000.0 * 100;
FLOAT c_discount;
MONEY c_balance = -10.0 * 100;
MONEY c_ytd_payment = 10.0 * 100;
COUNT c_payment_cnt = 1;
COUNT c_delivery_cnt = 0;
TEXT c_data[500+1];
TEXT c_data1[250+1];
TEXT c_data2[250+1];
ID   len;

int bulk_c;

void
LoadCustomer(w1, w2)
   ID w1, w2;
{
   ID w_id;

   print_prologue("Customer");
  bulk_c = bulk_open("tpcc", "customer", password, &blkdesc,
&sliceNum);
   for (w_id=w1; w_id<=w2; w_id++)
   {
      Customer(w_id);
      printf("\nLoaded customer for warehouse %d into
partition %d \n", w_id, sliceNum);
   }
   end_customer_load();
   return;
}
void
Customer(w_id)
   int w_id;
{
   BitVector badcredit[DIST_PER_WARE][(3000+WSZ-
1)/WSZ], * bmp;
   int i, j;
   ID d_id;

   /* Mark exactly 10% of customers as having bad credit */
   for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
   {
      bmp = badcredit[d_id-1];
      for (i=0; i<(3000+WSZ-1)/WSZ; i++)
         bmp[i] = (BitVector)0x0000;
      for (i=0; i<(3000+9)/10; i++)
      {
         do {
            j = RandomNumber(0,3000-1);
         } while (nthbit(bmp,j));
         setbit(bmp,j);
      }
   }

   c_w_id = w_id;
   for (i=0; i<CUST_PER_DIST; i++)
   {
      c_id = i+1;
      for (d_id=1;  d_id <= DIST_PER_WARE;  d_id++)
      {
         c_d_id = d_id;

   LastName(i<1000?i:NURandomNumber(255,NURAND_C,
0,999),c_last);
         MakeAlphaString(8, 16, c_first);

   MakeAddress(c_street_1,c_street_2,c_city,c_state,c_zip);
         MakeNumberString(16, 16, c_phone);
         MakeAlphaString(300, 500, c_data);
         datetime(&c_since);
         c_credit[0] = nthbit(badcredit[d_id-1],i) ? 'B' : 'G';
         c_discount = RandomNumber(0, 5000) / 10000.0;

         /* Break the string c_data into 2 pieces */
         len = strlen(c_data);
         if (len > 250)
```

```
        {
        memcpy(c_data1, c_data, 250);
        c_data1[250]='\0';
        memcpy(c_data2, c_data+250, len-250 +1);
        }

      else
        {
        memcpy(c_data1, c_data, len+1);
        strcpy(c_data2,"");
        }
      begin_customer_load();
      customer_load();
    }
  }
  return;
}
void
begin_customer_load()
{
  int  i = 1;


  bulk_bind(bulk_c, i++, "c_id",    &c_id,   ID_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_d_id",  &c_d_id, ID_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_w_id", &c_w_id, ID_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_first",   c_first, TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_middle",  c_middle,
TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_last",   c_last,   TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "street_1",  c_street_1 ,
TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "street_2",  c_street_2,
TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_city",   c_city,    TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_state",   c_state,
TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_zip", c_zip,     TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_phone",   c_phone,
TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_since",   &c_since,
DATE_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_credit",  c_credit,
  TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_credit_lim", &c_credit_lim,
MONEY_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_discount", &c_discount,
FLOAT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_delivery_cnt", &c_delivery_cnt,
COUNT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_payment_cnt",&c_payment_cnt,
COUNT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_balance", &c_balance,
MONEY_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_ytd_payment", &c_ytd_payment,
MONEY_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_data_1", c_data1,
TEXT_T,blkdesc);
  bulk_bind(bulk_c, i++, "c_data_2", c_data2,
TEXT_T,blkdesc);
  return;
}
void
customer_load()
{
  debug("c_id=%-5d  d_id=%-5d  w_id=%-5d  c_last=%s\n",
    c_id,  c_d_id,  c_w_id,  c_last);

  /* load the data */
  bulk_load(bulk_c, blkdesc);
  return;
}
void
end_customer_load()
{
  bulk_close(bulk_c, blkdesc);
  return;
}



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

Order, Order line, New order

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

/* Order row */
ID o_id;
ID o_c_id;
ID o_d_id;
ID o_w_id;
DATE o_entry_d;
ID o_carrier_id;
COUNT o_ol_cnt;
LOGICAL o_all_local;

/* Order line row */
ID   ol_o_id;
ID   ol_d_id;
ID   ol_w_id;
ID   ol_number;
ID   ol_i_id;
ID   ol_supply_w_id;
DATE ol_delivery_d;
COUNT ol_quantity;
MONEY ol_amount;
TEXT  ol_dist_info[24+1];

/* new order row */
ID no_o_id;
ID no_d_id;
ID no_w_id;


int  o_bulk;
int ol_bulk;
int no_bulk;


void
LoadOrd(w1, w2)
  ID w1, w2;
{
  ID w_id;
  ID d_id;

  print_prologue("Order");
  ol_bulk = bulk_open("tpcc", "order_line", password,
&blkdesc1, &sliceNum);
  o_bulk = bulk_open("tpcc", "orders", password, &blkdesc,
&sliceNum);
  for (w_id=w1; w_id<=w2; w_id++)
  {

    for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
      Orders(w_id, d_id);

    printf("\nLoaded order + order_line for warehouse %d\n",
w_id);
  }
  end_order_line_load();
  end_order_load();
  return;
}
void
LoadNew(w1, w2)
  ID w1, w2;
{
  ID w_id;
  ID d_id;

  print_prologue("NewOrder");
```

```
   no_bulk = bulk_open("tpcc", "new_order", password,
&blkdesc, &sliceNum);
   for (w_id=w1; w_id<=w2; w_id++)
   {
      for (d_id = 1;  d_id <= DIST_PER_WARE;  d_id++)
      {
         no_d_id = d_id;
         no_w_id = w_id;
         for (no_o_id=2101;  no_o_id <= ORD_PER_DIST;
no_o_id++)
         {
            begin_new_order_load();
            new_order_load();
         }
      }
      printf("\nLoaded new_order for warehouse %d\n", w_id);
   }
   end_new_order_load();
   return;
}
void
Orders(w_id, d_id)
   ID w_id;
   ID d_id;
{
   int cust[ORD_PER_DIST+1];
   int ol_cnt[ORD_PER_DIST+1], sum;
   ID ol;

   printf("\nLoading orders and order lines  for warehouse %d
district %d\n",
    w_id, d_id);

   RandomPermutation(cust, ORD_PER_DIST);


   for (o_id = 1, sum=0;  o_id <= ORD_PER_DIST;  o_id++)
            sum += (ol_cnt[o_id] = RandomNumber(5, 15));

      while (sum > 10*ORD_PER_DIST)
      {
            do {
                  o_id = RandomNumber(1,ORD_PER_DIST);
            } while (ol_cnt[o_id]==5);
            ol_cnt[o_id]--;
            sum--;
      }

      while (sum < 10*ORD_PER_DIST)
      {
            do {
                  o_id = RandomNumber(1,ORD_PER_DIST);
            } while (ol_cnt[o_id]==15);
            ol_cnt[o_id]++;
            sum++;
      }

   for (o_id = 1;  o_id <= ORD_PER_DIST;  o_id++)
   {
   o_c_id = cust[o_id];
   o_d_id = d_id;
   o_w_id = w_id;
   datetime(&o_entry_d);
   if (o_id <= 2100)
   o_carrier_id = RandomNumber(1,10);
   else o_carrier_id = -1;
   o_ol_cnt = ol_cnt[o_id];
   /* o_ol_cnt = RandomNumber(5, 15); */
   o_all_local = 1;
   begin_order_load();
   order_load();

   for (ol=1;  ol<=o_ol_cnt; ol++)
      OrderLine(ol);

   }
   return;
}



void
OrderLine(ol)
   ID ol;
{

   ol_o_id = o_id;
   ol_d_id = o_d_id;
   ol_w_id = o_w_id;
   ol_number = ol;
   ol_i_id = RandomNumber(1, MAXITEMS);
   ol_supply_w_id = o_w_id;
   if (o_id <= 2100)
      ol_delivery_d = o_entry_d;
   ol_quantity = 5;
   if (o_id <= 2100) ol_amount = 0;
   else       ol_amount = RandomNumber(1, 999999);
   MakeAlphaString(24, 24, ol_dist_info);
   begin_order_line_load();
   order_line_load();
   return;
}

void
NewOrder(w_id, d_id)
   ID w_id, d_id;
{

   no_d_id = o_d_id;
   no_w_id = o_w_id;
   for (no_o_id=2101;  no_o_id <= ORD_PER_DIST;
no_o_id++)
   {
      begin_new_order_load();
      new_order_load();
   }
   return;
}



void
begin_order_load()
{
   int  i = 1;


   bulk_bind(o_bulk, i++, "o_id", &o_id, ID_T,blkdesc);
   bulk_bind(o_bulk, i++, "o_c_id", &o_c_id, ID_T,blkdesc);
   bulk_bind(o_bulk, i++, "o_d_id", &o_d_id, ID_T,blkdesc);
   bulk_bind(o_bulk, i++, "o_w_id", &o_w_id, ID_T,blkdesc);
   bulk_bind(o_bulk, i++, "o_entry_d", &o_entry_d,
DATE_T,blkdesc);
   bulk_bind(o_bulk, i++, "o_carrier_id", &o_carrier_id,
ID_T,blkdesc);
   bulk_bind(o_bulk, i++, "o_ol_cnt", &o_ol_cnt,
COUNT_T,blkdesc);
   bulk_bind(o_bulk, i++, "o_all_local", &o_all_local,
LOGICAL_T,blkdesc);
   return;
}
void
order_load()
{
   debug("o_id=%d o_c_id=%d  count=%d\n", o_id, o_c_id,
o_ol_cnt);
   bulk_load(o_bulk, blkdesc);
   return;
}
void
end_order_load()
{
   bulk_close(o_bulk, blkdesc);
   return;
}
```

```
void
begin_order_line_load()
{
   int  i = 1;


   bulk_bind(ol_bulk, i++, "ol_o_id", &ol_o_id, ID_T,blkdesc1);
   bulk_bind(ol_bulk, i++, "ol_d_id", &ol_d_id, ID_T,blkdesc1);
  bulk_bind(ol_bulk, i++, "ol_w_id", &ol_w_id, ID_T,blkdesc1);
   bulk_bind(ol_bulk, i++, "ol_number", &ol_number,
ID_T,blkdesc1);
   bulk_bind(ol_bulk, i++, "ol_i_id", &ol_i_id, ID_T,blkdesc1);
   bulk_bind(ol_bulk, i++, "ol_supply_w_id", &ol_supply_w_id,
ID_T,blkdesc1);
   bulk_bind(ol_bulk, i++, "ol_delivery_d", &ol_delivery_d,
DATE_T,blkdesc1);
   bulk_bind(ol_bulk, i++, "ol_quantity", &ol_quantity,
COUNT_T,blkdesc1);
   bulk_bind(ol_bulk, i++, "ol_amount", &ol_amount,
MONEY_T,blkdesc1);
   bulk_bind(ol_bulk, i++, "ol_dist_info", ol_dist_info,
TEXT_T,blkdesc1);
   return;
}
void
order_line_load()
{
   debug("  ol_o_id=%d  ol_number=%d  ol_amount=%g\n",
       ol_o_id,  ol_number, ol_amount);
   bulk_load(ol_bulk, blkdesc1);
   return;
}

void
end_order_line_load()
{
   bulk_close(ol_bulk, blkdesc1);
   return;
}
void
begin_new_order_load()
{
   int  i = 1;


  bulk_bind(no_bulk, i++, "no_o_id", &no_o_id, ID_T,blkdesc);
  bulk_bind(no_bulk, i++, "no_d_id", &no_d_id, ID_T,blkdesc);
   bulk_bind(no_bulk, i++, "no_w_id", &no_w_id,
ID_T,blkdesc);
   return;
}
void
new_order_load()
{
   debug("  no_o_id=%d  \n", no_o_id);
   bulk_load(no_bulk, blkdesc);
   return;
}

void
end_new_order_load()
{
   bulk_close(no_bulk, blkdesc);
   return;
}


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

Stock

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


ID s_i_id;
ID s_w_id;
COUNT s_quantity;
TEXT s_dist_01[24+1];
TEXT s_dist_02[24+1];
TEXT s_dist_03[24+1];
TEXT s_dist_04[24+1];
TEXT s_dist_05[24+1];
TEXT s_dist_06[24+1];
TEXT s_dist_07[24+1];
TEXT s_dist_08[24+1];
TEXT s_dist_09[24+1];
TEXT s_dist_10[24+1];
COUNT s_ytd;
COUNT s_order_cnt;
COUNT s_remote_cnt;
TEXT s_data[50+1];

int bulk_s;

/*
** On loading stock in major order of item_id:
** 10% of the MAXITEMS items in each warehouse need to
marked as original
** (i.e., s_data like '%ORIGINAL%'.)  This is a bit harder to do
when we
** load by item number, rather than by warehouses.  The trick
is to first
** generate a huge WAREBATCH * MAXITEMS bitmap,
initialize all bits to zero,
** and then set 10% of bits in each row to 1.  While loading
item i in
** warehouse w, we simply lookup bitmap[w][i] to see whether
it needs to
** be marked as original.
*/

#ifdef _NTINTEL
/* On NT stack overflow happens when you define the
following on stack
*/
   BitVector original[WAREBATCH][((MAXITEMS+(WSZ-
1))/WSZ)] ;
#endif
void
LoadStock(w1, w2)
   ID w1, w2;
{
   ID w_id;


   int w, i, j;

   print_prologue_item("Stock");
   if (w2-w1+1 > WAREBATCH)
   {
      fprintf(stderr, "Can't load stock for %d warehouses.\n",
         w2-w1+1);
      fprintf(stderr, "Please use batches of %d.\n",
WAREBATCH);
   }

   for (w=w1; w<=w2; w++)
   {
      bmp = original[w-w1];
      /* Mark all items as not "original" */
      for (i=0; i<(MAXITEMS+(WSZ-1))/WSZ; i++)
         bmp[i] = (BitVector)0x0000;
      /* Mark exactly 10% of items as "original" */
      for (i=0; i<(MAXITEMS+9)/10; i++)
      {
         do {
            j = RandomNumber(0,MAXITEMS-1);
         } while (nthbit(bmp,j));
```

```
        setbit(bmp,j);
      }
    }

    printf("Loading stock for warehouse %d to %d.\n", w1, w2);
    bulk_s = bulk_open("tpcc", "stock", password, &blkdesc,
&sliceNum);

    /* do for each item */
    for (s_i_id=i1;   s_i_id <= i2; s_i_id++)
    {
      for (w_id=w1; w_id<=w2; w_id++)
      {
      /* Generate Stock Data */
        s_w_id = w_id;
        s_quantity = RandomNumber(10,100);
        MakeAlphaString(24, 24, s_dist_01);
        MakeAlphaString(24, 24, s_dist_02);
        MakeAlphaString(24, 24, s_dist_03);
        MakeAlphaString(24, 24, s_dist_04);
        MakeAlphaString(24, 24, s_dist_05);
        MakeAlphaString(24, 24, s_dist_06);
        MakeAlphaString(24, 24, s_dist_07);
        MakeAlphaString(24, 24, s_dist_08);
        MakeAlphaString(24, 24, s_dist_09);
        MakeAlphaString(24, 24, s_dist_10);
        s_ytd = 0;
        s_order_cnt = 0;
        s_remote_cnt = 0;
        MakeAlphaString(26, 50, s_data);
        if (nthbit(original[w_id-w1],s_i_id-1))
        {
          Original(s_data);
        }
        begin_stock_load();
        stock_load();
      }
    }
    end_stock_load();
    printf("\nLoaded stock for warehouses %d to %d.\n", w1,
w2);
    return;
}
void
begin_stock_load()
{
    int   i = 1;


    bulk_bind(bulk_s, i++, "s_i_id", &s_i_id, ID_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_w_id", &s_w_id, ID_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_quantity", &s_quantity,
COUNT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_ytd",  &s_ytd,
    COUNT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_order_cnt", &s_order_cnt,
COUNT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_remote_cnt", &s_remote_cnt,
COUNT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_01", s_dist_01,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_02", s_dist_02,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_03", s_dist_03,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_04", s_dist_04,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_05", s_dist_05,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_06", s_dist_06,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_07", s_dist_07,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_08", s_dist_08,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_09", s_dist_09,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_dist_10", s_dist_10,
TEXT_T,blkdesc);
    bulk_bind(bulk_s, i++, "s_data", s_data,  TEXT_T,blkdesc);
    return;
}



void
stock_load()
{
    debug("s_i_id=%d w_id=%d s_data=%s\n",
       s_i_id, s_w_id,   s_data);
    bulk_load(bulk_s, blkdesc);
    return;
}

void
end_stock_load()
{
    bulk_close(bulk_s, blkdesc);
    return;
}


void
test(){
    return;
}

void
getargs(argc, argv)

/*****************************************************************
configure configures the load stuff
   By default, loads all the tables for a the specified
warehouse.
        When loading warehouse 1, also loads the item table.
*****************************************************************/
    int argc;
    char **argv;
{
    if (argc == 1)
    {
      printf("Usage: %s <table> <w_first> [<w_last>] [<SA
passwd> [<num threasd> [no_partition]]] \n", argv[0]);
      exit(0);
    }

    /* define the defaults */
    load_item = load_warehouse = load_district = load_history
=
    load_orders = load_new_order = load_order_line =
    load_customer = load_stock = NO;


    if   (strcmp(argv[1], "warehouse") == 0) load_warehouse =
YES;
    else if (strcmp(argv[1], "district") == 0)  load_district = YES;
    else if (strcmp(argv[1], "stock") == 0)  load_stock = YES;
    else if (strcmp(argv[1], "item") == 0)  load_item = YES;
    else if (strcmp(argv[1], "history") == 0)  load_history = YES;
    else if (strcmp(argv[1], "orders") == 0)  load_orders = YES;
    else if (strcmp(argv[1], "customer") == 0)  load_customer =
YES;
    else if (strcmp(argv[1], "new_order") ==0) load_new_order
= YES;
    else if (strcmp(argv[1], "-v") ==0)
    {
      printf("Usage: %s <table> <w_first> [<w_last>] [<SA
passwd> [<num threasd> [no_partition]]] \n", argv[0]);
      exit(0);
    }
    else
    {
      printf("%s is not a valid table name\n", argv[1]);
      exit(0);
```

```
  }

  /* Set the w1 and w2 to argv[2] and arg[3] */
  if (argc < 3)
  {
     printf("Usage: %s <table> <w_first> [<w_last>] [<SA
passwd> [<num threasd> [no_partition]]] \n", argv[0]);
     exit(1);
  }
  {
     w1 = atoi(argv[2]);
     if (argc >= 3)
        w2 = atoi(argv[3]);
     else
        w2 = w1;
  }

  i1 = 1;
  i2 = MAXITEMS;
  /* Get the password for sa */
  if (argc > 4)
  strcpy(password,argv[4]);

  if (argc > 5)
  {
     numOfSlices = atoi(argv[5]);
  }
  else
  {
     numOfSlices = INVALID_SLICE_NUM;
     loadSlices = 0;
  }

  if( argc > 6 )
  {
     if(strcmp(argv[6], "no_partition") == 0)
     {
        loadSlices = 0;
     }
  }

  if(loadSlices)
  {
     printf (" Number of partition is %d \n", numOfSlices);
  }

  if(!loadSlices && (numOfSlices != INVALID_SLICE_NUM) )
  {

    printf(" Parallel loading with %d threads \n", numOfSlices);
  }
```

```
     /* Check if warehouse is within the range */
     if (w1 <= 0 || w1 > w2)
     {
        printf("Warehouse id is out of range\n");
        exit(0);
     }
     return;
}

#ifndef _NTINTEL
double drand48();
#endif
void
MakeAddress(str1, str2, city, state, zip)
     TEXT str1[20+1];
     TEXT str2[20+1];
     TEXT city[20+1];
     TEXT state[2+1];
     TEXT zip[9+1];
{
     MakeAlphaString(10,20,str1);
     MakeAlphaString(10,20,str2);
     MakeAlphaString(10,20,city);
     MakeAlphaString(2,2,state);
     MakezipString(0,9999,zip);

     /* Changed for TPCC V 3.0 */
     strcat(zip, "11111");

     return;
}


void
LastName(num, name)
/*****************************************************************
Lastname generates a lastname from a number.
*****************************************************************/
     int num;
     char name[20+1];
{
  static char *n[] = {"BAR", "OUGHT", "ABLE", "PRI", "PRES",
                      "ESE", "ANTI", "CALLY", "ATION", "EING"};

     strcpy(name, n[(num/100)%10]);
     strcat(name, n[(num/10) %10]);
     strcat(name, n[(num/1)  %10]);
     return;
}


int MakeNumberString(min, max, num)
```

```
     int min;
     int max;
     TEXT num[];
{
     static char digit[]="0123456789";
     int length;
     int i;

     length = RandomNumber(min, max);

     for (i=0;  i<length;  i++)
     num[i] = digit[RandomNumber(0,9)];
     num[length] = '\0';

     return length;
}

int MakezipString(min, max, num)
        int min;
        int max;
        TEXT num[];
{
        static char digit[]="0123456789";
        int length;
        int i;

        length = 4;

        for (i=0;  i<length;  i++)
        num[i] = digit[RandomNumber(0,9)];
        num[length] = '\0';

        return length;
}



int MakeAlphaString(min, max, str)
     int min;
     int max;
     TEXT str[];
{
     static char character[] =
     "abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRS
TUVWXYZ0123456789";
     int length;
     int i;

     length = RandomNumber(min, max);

     for (i=0;  i<length;  i++)
     str[i] = character[RandomNumber(0, sizeof(character)-2)];
```

```
   str[length] = '\0';

   return length;
}

void
Original(str)
   TEXT str[];
{
   int pos;
   int len;

   len = strlen(str);
   if (len < 8) return;

   pos = RandomNumber(0,len-8);

   str[pos+0] = 'O';
   str[pos+1] = 'R';
   str[pos+2] = 'I';
   str[pos+3] = 'G';
   str[pos+4] = 'I';
   str[pos+5] = 'N';
   str[pos+6] = 'A';
   str[pos+7] = 'L';
   return;
}



void
RandomPermutation(perm, n)
   int perm[];
   int n;
{
   int i, r, t;

   /* generate the identity permutation to start with */
   for (i=1; i<=n; i++)
   perm[i] = i;

   /* randomly shuffle the permutation */
   for (i=1; i<=n; i++)
   {
   r = RandomNumber(i, n);
   t = perm[i]; perm[i] = perm[r]; perm[r] = t;
   }
   return;
}
```

```
void
Randomize()
{
#ifdef _NTINTEL
   srand(time(0)+_getpid());
#else
   srand48(time(0)+getpid());
#endif
   return;
}


int
RandomNumber(int min, int max)
{
   int r;
#ifdef _NTINTEL
   r = (int)((float)rand()/(float)(RAND_MAX + 1) * (max - min +
1)) + min;
#else
   r = (int)(drand48() * (max - min + 1)) + min;
#endif
   return r;
}


int
NURandomNumber(int a, int c, int min, int max)
{
   int r;

   r = ((RandomNumber(0, a) | RandomNumber(min, max)) +
c)
   % (max - min + 1) + min;

   return r;
}

void
donothing(const char *fmt, ...)
{
   return;
}
```

tpcc_ld_error.c

```
#include "stdio.h"
#include "ctpublic.h"
#ifdef _NTINTEL
#include <stdlib.h>
#include <windows.h>
#endif

CS_INT  cl_err_handler(context, connection, errmsg)
CS_CONTEXT      *context;
CS_CONNECTION   *connection;
CS_CLIENTMSG    *errmsg;

{
   fprintf(stderr, "Open Client Error (%d, %d, %d, %d)\n",
         CS_LAYER(errmsg->msgnumber),
         CS_ORIGIN(errmsg->msgnumber),
         CS_SEVERITY(errmsg->msgnumber),
         CS_NUMBER(errmsg->msgnumber));

   fprintf(stderr, "%s\n", errmsg->msgstring);

   if (errmsg->osstringlen > 0)
   {
      fprintf(stderr, "Operating System Error:\n%s\n",
         errmsg->osstring);
   }

   return (CS_SUCCEED);
}   /* end of cl_err_handler () */

CS_INT server_msg_handler(context, connection, srvmsg)
CS_CONTEXT     *context;
CS_CONNECTION  *connection;
CS_SERVERMSG   *srvmsg;

{
   if (srvmsg->msgnumber != 5701)
   {
      fprintf(stderr, "Server message %ld, Severity %ld, State
%ld\n",
         srvmsg->msgnumber, srvmsg->severity, srvmsg-
>state);

      if (srvmsg->svrnlen > 0)
      {
         fprintf(stderr, "Server '%s' ", srvmsg->svrname);
      }

      if (srvmsg->proclen > 0)
      {
         fprintf(stderr, "Procedure '%s' ", srvmsg->proc);
      }

      if (srvmsg->line > 0)
      {
```

```
        fprintf(stderr, "Line '%d' ", srvmsg->line);
      }

      fprintf(stderr, "\n %s\n", srvmsg->text);
   }

   return(CS_SUCCEED);
}  /* end of server_msg_handler()  */

CS_INT cs_err_handler(context, errmsg)
CS_CONTEXT      *context;
CS_CLIENTMSG    *errmsg;
{
   fprintf(stderr, "CS Lib Error %d:\n%s\n",
         errmsg->msgnumber, errmsg->msgstring);

   return (CS_SUCCEED);
}  /* end of cs_err_handler() */
```

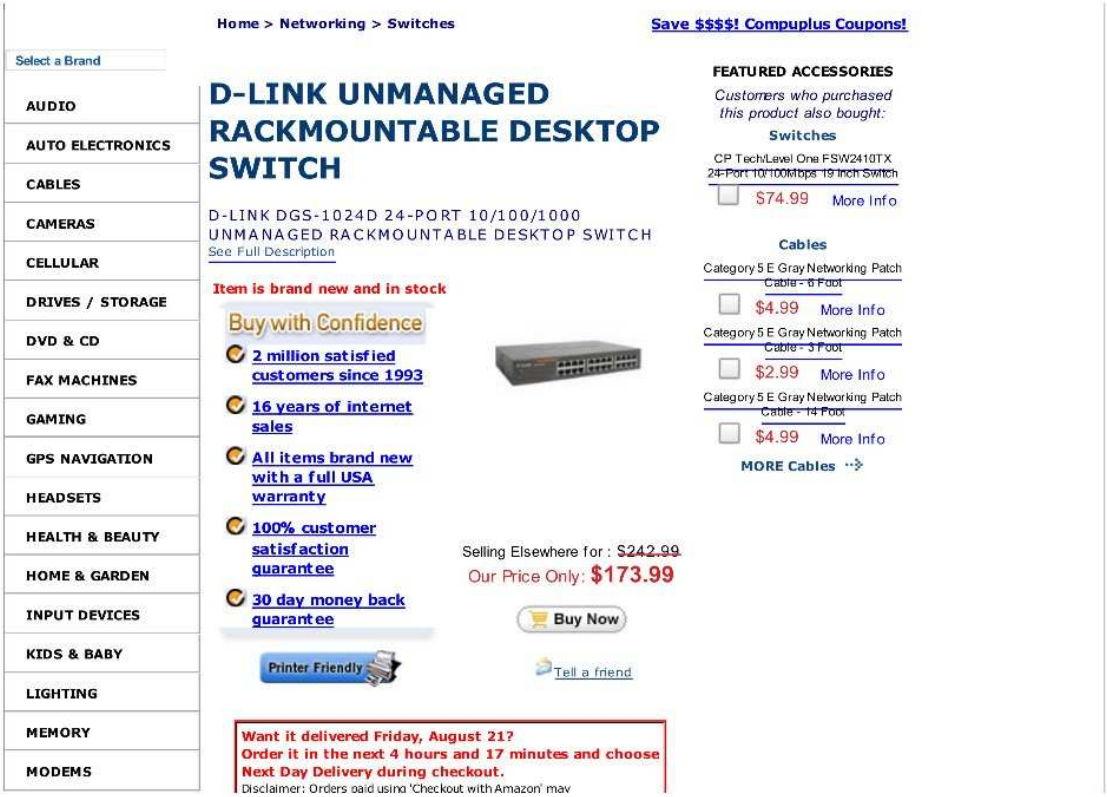
tpcc_ld_bulk_sybase.c

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

Sybase Specific Routines

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

#include <stdio.h>
#include <stdlib.h>
#include <ctpublic.h>
#include <bkpublic.h>
#include <unistd.h>
#ifdef _NTINTEL
#include <sys/timeb.h>
#include <io.h>
#else
#include <sys/time.h>
#endif
#include <string.h>
#include "tpcc_ld_loader.h"
void
datetime(date)
DBDATETIME *date;
{
#ifdef _NTINTEL
   time_t time1;
   time(&time1);
   date->dtdays = time1 / (60*60*24)
         + (1970-1900)*365 + (1970-1900)/4;
   date->dttime = (time1 % (60*60*24))*300 ;
#else
   struct timeval time;
   gettimeofday(&time, NULL);
   date->dtdays = time.tv_sec / (60*60*24)
         + (1970-1900)*365 + (1970-1900)/4;
   date->dttime = (time.tv_sec % (60*60*24))*300
         + time.tv_usec*300/1000000;
#endif
return;
}


/* define the type information for each field */
typedef struct
{
   char *terminator;
   int termlen;
   int type;
   int cstype;
} bind_parm;

bind_parm parm[MAX_T] =
{
   /* COUNT */   {NULL, 0, CS_INT_TYPE, SYBINT4},
   /* ID   */ {NULL, 0, CS_INT_TYPE, SYBINT4},
   /* MONEY */   {NULL, 0, CS_FLOAT_TYPE, SYBFLT8},
   /* FLOAT */ {NULL, 0, CS_FLOAT_TYPE, SYBFLT8},
       /* TEXT */      {"",   1, CS_CHAR_TYPE, SYBCHAR},
   /* DATE */   {NULL, 0, CS_DATETIME_TYPE,
SYBDATETIME},
   /* LOGICAL */{NULL, 0, CS_INT_TYPE, SYBINT4}
};

#define MAXOPENS 10

#define RETURN_IF(a,b)                  \
   if (a != CS_SUCCEED)                 \
   {                                    \
      fprintf(stderr, "Error in: %s\n", b);   \
      return a;                         \
   }

#define EXIT_IF(a) if (a != CS_SUCCEED)\
   { fprintf(stderr, "FATAL ERROR! Line %d\n", __LINE__);
exit(-1);}

CS_CONNECTION *dbconn[MAXOPENS];
int   count[MAXOPENS];

CS_RETCODE    init_db();
CS_RETCODE    connect_db();
CS_RETCODE    send_sql();
void          handle_returns();

extern  CS_INT   server_msg_handler();
extern  CS_INT   cl_err_handler();
extern  CS_INT   cs_err_handler();


CS_CONTEXT    *cntx_ptr;
int bulk_open(database, table, password, blk_desc, sliceNum)
   CS_CHAR    *database;
   CS_CHAR    *table;
   CS_CHAR    *password;
   CS_BLKDESC **blk_desc;
   CS_INT   *sliceNum;
{
   int db;

   CS_RETCODE     retcode=0;
   CS_CONNECTION *conn_ptr;
   CS_COMMAND    *cmd_ptr;

   char    tblname[25];

   /* make note we have established a connection */
   for (db=0; db<MAXOPENS; db++)
      if (dbconn[db] == NULL) break;
   count[db] = 0;

   retcode = CS_SUCCEED;
   retcode = init_db(&cntx_ptr);
   EXIT_IF(retcode);

   strcpy(tblname, table);
   retcode = connect_db(cntx_ptr, &conn_ptr, "sa", "", table);
   EXIT_IF(retcode);

   retcode = ct_cmd_alloc(conn_ptr, &cmd_ptr);
   EXIT_IF(retcode);

   retcode = send_sql(cmd_ptr, "use tpcc");
   EXIT_IF(retcode);

   handle_returns(cmd_ptr);
   /* prepare to do a bulk copy */
   retcode = blk_alloc(conn_ptr, CS_CURRENT_VERSION,
blk_desc);
   EXIT_IF(retcode);

   if ( *sliceNum != INVALID_SLICE_NUM )
   {
```

```
      retcode =  retcode = blk_props(*blk_desc, CS_SET,
BLK_SLICENUM,
                        sliceNum, CS_UNUSED, NULL) ;
            EXIT_IF(retcode);
   }

   retcode = blk_init(*blk_desc, CS_BLK_IN,
         tblname, strlen(tblname));
   EXIT_IF(retcode);

   return db;
}

CS_RETCODE
init_db(cntx_ptr)
CS_CONTEXT **cntx_ptr;
{
   CS_RETCODE    retcode=CS_SUCCEED;

   retcode = cs_ctx_alloc(CS_CURRENT_VERSION,
cntx_ptr);
   RETURN_IF(retcode, "init_db: cs_ctx_alloc");

   retcode = ct_init(*cntx_ptr, CS_CURRENT_VERSION);
   RETURN_IF(retcode, "init_db: ct_init");

   retcode = ct_callback(*cntx_ptr, NULL, CS_SET,
         CS_SERVERMSG_CB, (CS_VOID
*)server_msg_handler);
   RETURN_IF(retcode, "init_db: ct_callback-server_msg");

   retcode = cs_config(*cntx_ptr, CS_SET,
CS_MESSAGE_CB,
         (CS_VOID *)cs_err_handler, CS_UNUSED, NULL);
   RETURN_IF(retcode, "init_db: cs_config-cs_err");

   retcode = ct_callback(*cntx_ptr, NULL, CS_SET,
CS_CLIENTMSG_CB,
         (CS_VOID *)cl_err_handler);
   RETURN_IF(retcode, "init_db: ct_callback- cl_err");
   return retcode;
} /* end of init_db() */

CS_RETCODE
connect_db(cntx_ptr, conn_ptr, user_name, password, table)
CS_CONTEXT     *cntx_ptr;
CS_CONNECTION  **conn_ptr;
CS_CHAR        *user_name;
CS_CHAR        *password;
CS_CHAR        *table;
{
   CS_RETCODE    retcode=0;
   CS_BOOL        bool;
   CS_INT         packetsize=4096;

   retcode = ct_con_alloc(cntx_ptr, conn_ptr);
   RETURN_IF(retcode, "connect_db: ct_con_alloc");

   retcode = ct_con_props(*conn_ptr, CS_SET,
CS_USERNAME,
         user_name, CS_NULLTERM, NULL);
   RETURN_IF(retcode, "connect_db: ct_con_props");

   retcode = ct_con_props(*conn_ptr, CS_SET,
CS_PASSWORD,
         password, CS_NULLTERM, NULL);
   RETURN_IF(retcode, "connect_db: ct_con_props");

   retcode = ct_con_props(*conn_ptr, CS_SET,
CS_APPNAME,
         table, CS_NULLTERM, NULL);
   RETURN_IF(retcode, "connect_db: ct_con_props");

   retcode = ct_con_props(*conn_ptr, CS_SET,
CS_PACKETSIZE,
         &packetsize, CS_UNUSED, NULL);
   RETURN_IF(retcode, "connect_db: ct_con_props");

   bool = CS_TRUE;
   retcode = ct_con_props(*conn_ptr, CS_SET,
CS_BULK_LOGIN,
         &bool, CS_UNUSED, NULL);
   RETURN_IF(retcode, "connect_db: ct_con_props");

   /*establish a connection with the server specified by
DSQUERY */
   retcode = ct_connect(*conn_ptr, NULL, 0);
   RETURN_IF(retcode, "connect_db: ct_connect");
   return retcode;
}  /* connect_db() */

CS_RETCODE
send_sql(cmd_ptr, sqltext)
CS_COMMAND    *cmd_ptr;
CS_CHAR       *sqltext;
{
   CS_RETCODE    retcode=0;

   retcode = ct_command(cmd_ptr, CS_LANG_CMD, sqltext,
         CS_NULLTERM, CS_UNUSED);
   RETURN_IF(retcode, "send_sql:ct_command");

   retcode = ct_send(cmd_ptr);
   RETURN_IF(retcode, "send_sql:ct_send");
   return retcode;
}  /* end of send_sql()  */

void
handle_returns(cmd_ptr)
CS_COMMAND   *cmd_ptr;
{
   CS_INT        result_type;

  while (ct_results(cmd_ptr, &result_type) == CS_SUCCEED)
   {
      ;
   }
   return;
}  /* end of handle_returns()  */


CS_DATAFMT    datafmt[255];
CS_INT        datalen[255];

CS_RETCODE
bulk_bind(int db, int column, char *name, void *address, int
type, CS_BLKDESC *blk_desc)
{
   CS_RETCODE    retcode=0;


   datafmt[column].locale = 0;
   datafmt[column].count  = 1;
   datafmt[column].datatype = parm[type].type;

   if (parm[type].cstype == SYBCHAR)
   {
      datafmt[column].maxlength = 255;
            datalen[column] = strlen(address);
   }
   else
   {
      datafmt[column].maxlength = sizeof(CS_INT);
      datalen[column] = CS_UNUSED;
   }

   retcode = blk_bind(blk_desc, column, &datafmt[column],
         address, &datalen[column], NULL);
   RETURN_IF(retcode, "bulk_bind: blk_bind");
   return retcode;
}

/*
void
bulk_null(db, column)
   int db;
```

```
   int column;
{
   if (bcp_collen(dbproc[db], 0, column) != SUCCEED)
      printf("Can't null column %d\n", column);
}



void
bulk_non_null(db, column)
   int db;
   int column;
{
   if (bcp_collen(dbproc[db], -1, column) != SUCCEED)
      printf("Can't non-null column %d\n", column);
}
*/

/*void */
CS_RETCODE
bulk_load(db, blk_desc)
   int db;
   CS_BLKDESC  *blk_desc;
{
   CS_RETCODE retcode=CS_SUCCEED;
   CS_INT     numofrows=0;

   count[db]++;
   retcode = blk_rowxfer(blk_desc);
   RETURN_IF(retcode, "bulk_load: blk_rowxfer");

   if (count[db]%batch_size == 0  )
   {
      retcode = blk_done(blk_desc, CS_BLK_BATCH,
&numofrows);
      if (retcode == CS_FAIL)
      {
         printf("bulk_load: Can't post rows\n");
         RETURN_IF(retcode, "bulk_load:blk_done");
      }
   }

#ifdef _NTINTEL
   if (count[db]%1000 == 0) _write(1,".",1);
   if (count[db]%50000 == 0) _write(1,"\n",1);
#else
   if (count[db]%1000 == 0) write(1,".",1);
   if (count[db]%50000 == 0) write(1,"\n",1);
#endif

   RETURN_IF(retcode, "bulk_load:blk_done");
   return retcode;
}


void
bulk_close(db, blk_desc)
int db;
CS_BLKDESC *blk_desc;
{
   CS_INT numofrows=0;

   if (blk_done(blk_desc, CS_BLK_ALL, &numofrows) ==
CS_FAIL)
      printf("Problems completing the bulk copy.\n");

   if (blk_drop(blk_desc) == CS_FAIL)
   {
      printf("blk_drop failed\n");
   }

   dbconn[db] = NULL;
   if (count[db] >= 1000) write(1,"\n",1);
   return;
}
```

13 Appendix D Pricing



Home > Networking > Switches

Save $$$$! Compuplus Coupons!

Select a Brand

- AUDIO
- AUTO ELECTRONICS
- CABLES
- CAMERAS
- CELLULAR
- DRIVES / STORAGE
- DVD & CD
- FAX MACHINES
- GAMING
- GPS NAVIGATION
- HEADSETS
- HEALTH & BEAUTY
- HOME & GARDEN
- INPUT DEVICES
- KIDS & BABY
- LIGHTING
- MEMORY
- MODEMS

D-LINK UNMANAGED RACKMOUNTABLE DESKTOP SWITCH

D-LINK DGS-1024D 24-PORT 10/100/1000 UNMANAGED RACKMOUNTABLE DESKTOP SWITCH

See Full Description

Item is brand new and in stock

Buy with Confidence

- 2 million satisfied customers since 1993
- 16 years of internet sales
- All items brand new with a full USA warranty
- 100% customer satisfaction guarantee
- 30 day money back guarantee

Printer Friendly

Selling Elsewhere for : $242.99

Our Price Only: $173.99

Buy Now

Tell a friend

Want it delivered Friday, August 21?
Order it in the next 4 hours and 17 minutes and choose Next Day Delivery during checkout.
Disclaimer: Orders paid using 'Checkout with Amazon' may

FEATURED ACCESSORIES

Customers who purchased this product also bought:

Switches

CP Tech/Level One FSW2410TX 24-Port 10/100Mbps 19 Inch Switch
$74.99 More Info

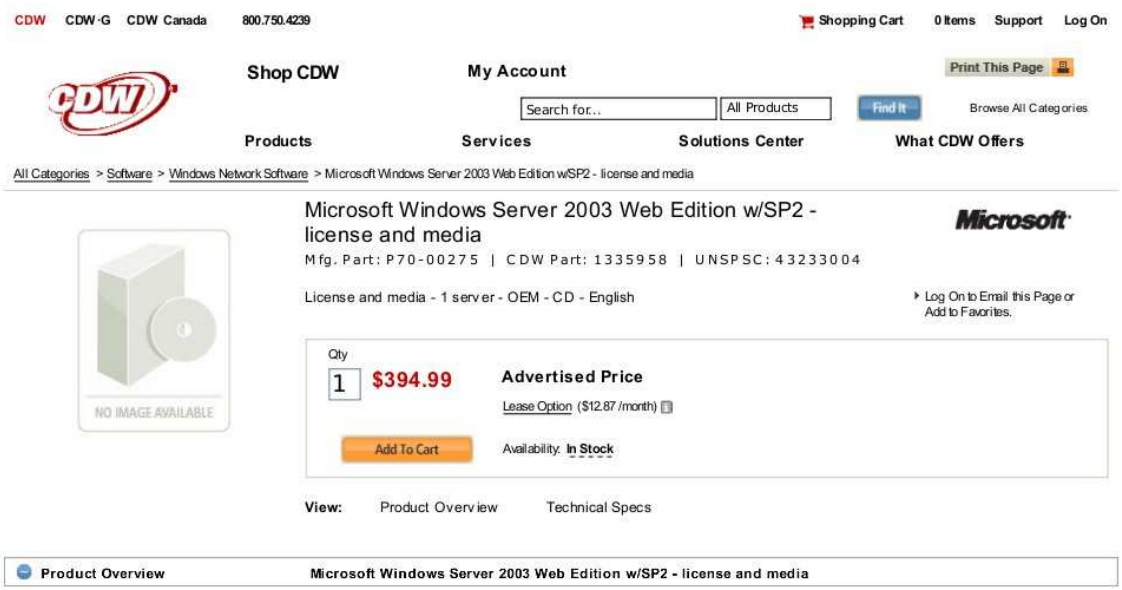
Cables

Category 5 E Gray Networking Patch Cable - 6 Foot
$4.99 More Info

Category 5 E Gray Networking Patch Cable - 3 Foot
$2.99 More Info

Category 5 E Gray Networking Patch Cable - 14 Foot
$4.99 More Info

MORE Cables


CDW CDW·G CDW Canada 800.750.4239
Shopping Cart 0 Items Support Log On
Shop CDW
My Account
Print This Page
Search for...
All Products
Find It
Browse All Categories
Products
Services
Solutions Center
What CDW Offers
All Categories > Software > Windows Network Software > Microsoft Windows Server 2003 Web Edition w/SP2 - license and media
Microsoft Windows Server 2003 Web Edition w/SP2 - license and media
Mfg. Part: P70-00275 | CDW Part: 1335958 | UNSPSC: 43233004
Microsoft
NO IMAGE AVAILABLE
License and media - 1 server - OEM - CD - English
Log On to Email this Page or Add to Favorites.
Qty
1
$394.99
Advertised Price
Lease Option ($12.87 /month)
Add To Cart
Availability: In Stock
View: Product Overview Technical Specs
Product Overview
Microsoft Windows Server 2003 Web Edition w/SP2 - license and media

- View Saved Carts
- Renew Subscriptions
- Store Catalog

Shopping Cart

Item	Quantity		Price	Line Total
New Subscription Contract August 20, 2009 - August 19, 2010				
Red Hat Enterprise Linux, Premium (Up to 2 Sockets for POWER) August 20, 2009 - August 19, 2010	1	Remove	$1,299.00	$1,299.00
Promotion Code:		Update Cart	**Subtotal:**	**$1,299.00**

Optional Install Discs and Documentation

add to cart Red Hat Enterprise Linux 5 (for POWER) $25 Media Kit (DVD only)

Continue shopping **Continue to Checkout**

ABOUT SSL CERTIFICATES

Home > Computer Hardware > Servers > UPS > Powercom > Item#:N82E16842106115

powercom King Pro KIN-1500AP 1500VA 900 Watts 5 Outlets UPS - Retail

Original Price: $149.99
You Save: $15.00
$134.99

ADD TO CART
ADD TO WISH LIST
EMAIL THIS PAGE
PRINT THIS PAGE
PRICE ALERT

Image Viewer

Protect Your Investment

Avoid the headaches of out-of warranty repairs; get the ultimate protection for your purchase!
(expand for options)

Special Offers

No interest for up to 12 months. Minimum purchase required.
Subject to credit approval. Details

No Payments for 90 Days on orders over $250.

OVERVIEW | CUSTOMER REVIEWS | SPECIFICATIONS | PRODUCT TOUR

Catastrophic losses of data, productivity, sales and image can all result from unstable power. As such, it's a potential cause of major financial loss for all kinds of companies and organizations whose business depends upon computers, information technology and other critical infrastructure. The Powercom King Pro series UPS provides complete protection for Internet computers and advanced servers.

It utilizes line interactive digital technology with advanced noise filtering, buck and boost, control and tel/modem surge suppression. This UPS provides precise, clean and stable computer grade power at all times. That powerful protector is the guardian that will shield against lightning, electrical noise, over/under voltage, brownouts and blackouts.

Review Summary (read more reviews, write a review)

5	73%	Product Rating:
4	9%	Total Reviews: 11
3	9%	
2	0%	
1	9%	

Not what i hoped for

Reviewed By: General on 5/4/2006
Tech Level: high - Ownership: 1 month to 1 year
This user purchased this item from Newegg

Pros: its an ok ups. it has the really loud beep and starts automatically like its supposed to but i noticed that running 3/4 capacity it held up for less an hour. unless i misunderstood the information, it should last 80 minutes at full load...

SYBASE

IBM Corporation
Mr. Steve Pratt
11501 Burnet Road
Austin, TX 78758

August 20, 2009

Dear Mr. Pratt,

Here is the information you requested regarding pricing of Sybase ASE 15.0.3 products used in the TPC-C benchmark.

Description	Unit Price	Quantity	Amount
ASE 15.0.3 Enterprise Edition for Linux on POWER (includes a 25% Adjustment for Multi-core) Catalog # 14023	$ 33,203	4	$ 132,810
SupportNow Standard Plan (includes a 25% Adjustment for Multi-core) Catalog # 98480	$ 29,217	3	$ 87,651
Subtotal			**$ 220,461**
Less Standard Price Discount*	15%		$ (33,069)
Total 3-year Cost			**$ 187,392**

** This discount level currently not available through eShop. Please contact Sybase Sales to obtain this discount.*

The total 3-year cost is US$ 187,392. The products are orderable through Sybase's normal distribution channels with the exception of eShop. Prices are valid for the next 90 days. If we can be of any further assistance, please contact David Bagley at dbagley@sybase.com.

Thank you!

Sybase, Inc.
One Sybase Drive
Dublin, CA. USA
94568
T: +1 925-236-5000
F: +1 925-236-4321

International Business Machines Corporation

11501 BURNETT RD
AUSTIN TX 78758

August 21, 2009

Below is the updated quote requested for the IBM Power 550 Express System and IBM System Storage Products.

Server Hardware	Part Number	Price Source	Unit Price	Qty	Ext. Price	3 -Yr. Maint
IBM Power 550 Express	8204-E8A5	1	5,509	1	5,509	
Op Panel Cable/Rk-m Drawer w/3.5 DASD	1877	1	6	1	6	
73.4GB 15K RPM SAS Disk Drive	3646	1	498	2	996	
16384MB (2x8192MB) RDIMMs, 400 MHz	4524	1	13,926	8	111,408	
2-core 4.2 GHz POWER6 Processor Card	4966	1	11,828	2	23,656	
One Processor Act for Proc Feature #4966	4986	1	7,573	2	15,146	
Dual-Port 1Gb Integrated Virt Eth Daughter Card	5623	1	301	1	301	
IDE Slimline DVD-ROM Drive	5756	1	208	1	208	
4 Gb Dual-Port Fibre Channel PCI-X 2.0 DDR	5759	1	2,499	2	4,998	
2-Port 10/100/1000 Base-TX Eth PCI Exp Adpt	5767	1	528	1	528	
4 Gigabit PCI Express Dual Port FC Adpt	5774	1	2,499	2	4,998	
Power Cable -- Drawer to IBM PDU, 14-foot	6458	1	14	1	14	
IBM/OEM Rack-Mount Drawer Rail Kit	7146	1	300	1	300	
IBM Rack-mount Drawer Bezel and Hardware	7360	1	450	1	450	
Power Supply, 1700 Watt AC, Hot-swap, Base	7707	1	699	1	699	
DASD/Media Bkpfor 3.5 DASD/DVD/Tape	8341	1	340	1	340	
Deskside Hardware Management Console	7042-C06	1	1,830	1	1,830	
T117 Color Display	3645	1	875	1	875	
IBM Quiet Touch USB Keyboard	5951	1	104	1	104	
24x7x4 3 Year Maintenance		1				7540
IBM System Storage EXP3000	1727-HC1	1	3,199	30	95,970	
IBM 1M SAS cable	39R6529	1	119	60	7,140	
IBM DS3000 (ESM)	39R6515	1	999	30	29,970	
IBM System Storage DS3400 Express	1726-42X	1	9,295	11	102,245	
IBM Hot-Swap 3.5 inch 73.4GB 15K SAS HDD	43W7523	1	309	488	150,792	
IBM TotalStorage SAN16B-2 Express Model	200516B	1	4,120	2	8,240	
IBM 4 Gbps SW SFP Transceiver	22R4902	1	150	21	3,150	
IBM 5m LC-LC Fiber Cable	39M5697	1	129	21	2,709	
16B Four Port Activation	22R4901	1	1,670	4	6,680	
IBM S2 42U Standard Rack	93074RX	1	1,489	2	2,978	
DS3000 Linux on Power Host License	44W2101	1	1,000	11	11,000	
ServicePac for 3-Year 24x7x4 Support (DS3400)	44J8073	1	1,300	11		14,300
ServicePac for 3-Year 24x7x4 Support (EXP3000)	41L2768	1	760	30		22,800
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	1	300	2		600
IBM System x3350 (Quad Xeon x3360 2.83GHz	419284U	1	1,869	8	14,952	
73.4GB 15K 2.5" SAS Hot-Swap Drive	43X0837	1	369	8	2,952	
ServicePac for 3-Year 24x7x4 Support (x3350)	51J9100	1	700	8		5,600
					Total	$661,984
					Total IBM Discount	$154,904
					Three-Year Cost of Ownership USD	$507,080

For more information on:
All products: http://www.ibm.com/products
For System p products: http://www-03.ibm.com/systems/p
For System x products: http://www-03.ibm.com/systems/x
For Storage products: http://www-03.ibm.com/systems/storage/disk/index.html

For additional information, please contact me directly:
Dan Hebrank
IBM Sales & Distribution, STG Sales
1-314-252-4160

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

May 13, 2008

IBM Corporation
Andrea Davis
11501 Burnet Road
Austin, TX 78758

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
127-00012	**Visual Studio Standard 2005** *Full License* *No Discount Applied*	$250	1	$250
N/A	**Microsoft Problem Resolution Services** *Professional Support* *(1 Incident)*	$245	1	$245

All products are currently orderable through Microsoft's normal distribution channels. A list of Microsoft's resellers can be found at **http://www.microsoft.com/products/info/render.aspx?view=22&type=mnp&content=22/licensing**

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at $245 per call.

This quote is valid for the next 90 days.

If we can be of any further assistance, please contact Jamie Reding at (425) 703-0510 or jamiere@microsoft.com.

Reference ID: PCAnDa0805130000008633.
Please include this Reference ID in any correspondence regarding this price quote.