
IBM eServer p5 570

Model 9117-570

Using

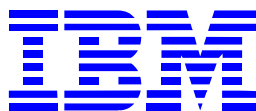
AIX 5L Version 5.3

and

Oracle Database 10g Enterprise Edition

TPC BenchmarkTM C

Full Disclosure Report



First Edition

October 13, 2005

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 eServer pSeries

IBM eServer xSeries

AIX

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

Microsoft Windows 2000 server and COM+ are registered trademarks of Microsoft Corporation

Oracle and Oracle Database 10g Standard Edition are registered trademarks of Oracle Corporation

First Edition: October 13, 2005

The information contained in this document is distributed on an AS IS basis without any warranty either expressed or implied. The use of this information or the implementation of any of these techniques is a customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used.

It is possible that this material may contain references to, or information about, IBM products (machines and programs), programming, or services that are not announced in your country. Such references or information must not be construed to mean that IBM intends to announce such products, programming, or services in your country.

All performance data contained in this publication was obtained in a controlled environment, and therefore the results which may be obtained in other operating environments may vary significantly. Users of this document should verify the applicable data in their specific environment.

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

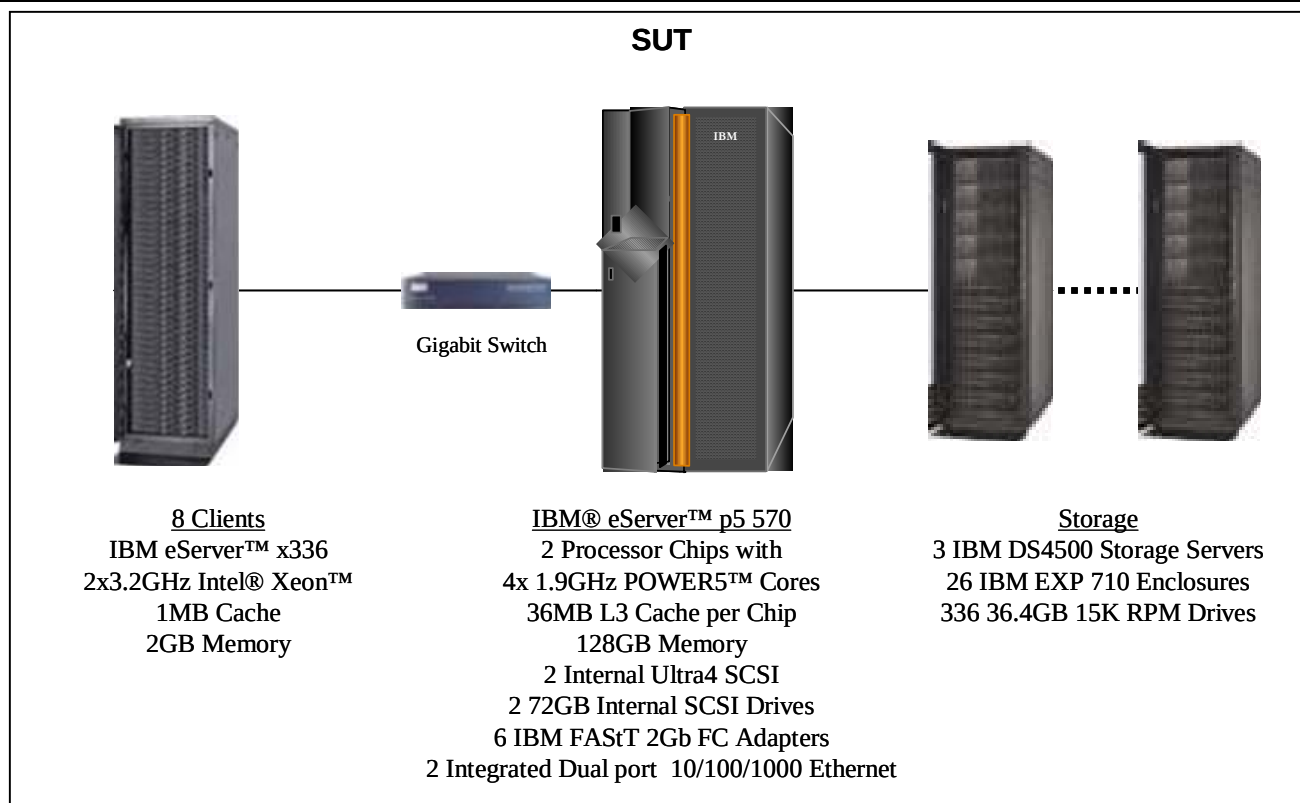
TPC Benchmark Administrator
IBM Commercial Performance
Mail Stop 9571
11501 Burnet Road
Austin, TX 78758
FAX Number (512) 838-1852

© **Copyright International Business Machines Corporation, 2005. All rights reserved.**

Permission is hereby granted to reproduce this document in whole or in part, provided the copyright notice printed above is set forth in full text on the title page of each item reproduced.

NOTE: US. Government Users - Documentation related to restricted rights: Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

 		IBM eServer p5 570 Model 9117-570		TPC-C Rev. 5.4
				Report Date: October 13, 2005
Total System Cost	TPC-C Throughput	Price/Performance		Availability Date
\$799,990 USD	203,439.87	\$3.93 USD		October 14, 2005
Processor Chips/Cores/Threads	Database Manager	Operating System	Other Software	No. Users
2/4/8	Oracle Database 10g Enterprise Edition	AIX 5L V5.3	Microsoft Visual C++ Microsoft COM+	162,560



System Components	Each of the 8 Clients		Server	
	Quantity	Description	Quantity	Description
Processors	2/2/2	3.2GHz 1MB L3 Xeon Processor	2/4/8	1.9GHz POWER5™ 36MB L3 Cache
Memory	2	1024MB	4	32GB
Disk Controllers	1	Ultra320 SCSI	2 6 3	Integrated Ultra4SCSI FAStT 2Gb FC Adapters IBM DS4500 Controllers
Disk Drives	1	36 GB	338 2	36.4GB 15K RPM FC 73.4GB 15K RPM FC
Total Storage		288GB		11,290 GB
Terminals	1	System Console	1	System Console

**ORACLE****IBM eServer p5 570
Model 9117-570****TPC-C Rev. 5.4****Report Date: October 13, 2005**

Description	Part No.	Src	Unit Price	Qty	Ext Price	Maint Price
Server Hardware						
Server 1:9117 Model 570	9117-570	1	2,922	1	2,922	7,146
Op Panel	1846	1	150	1	150	
IDE Slimline DVD-ROM Drive	2640	1	378	1	378	
73.4 GB 10,000 RPM Ultra320 SCSI Disk Drive	3274	1	439	2	878	
32768MB (4x8192MB) DIMMs, 208-pin, 200MHz	4492	1	103,000	4	412,000	
2 Gigabit Fibre Channel PCI-X Adapter	5716	1	2,267	6	13,602	
Power Cable -- Drawer to IBM PDU, 14-foot, 250V/10A	6458	1	14	2	28	
IBM Rack-mount Drawer Rail Kit	7164	1	168	1	168	
Processor, 0/2 Way 1.9 GHz with DDR1 memory	7832	1	10,900	2	21,800	7,680
CEC Backplane	7865	1	1,200	1	1,200	
I/O Backplane	7866	1	4,100	1	4,100	
Midplane	7867	1	500	1	500	
DASD Backplane	7868	1	1,200	1	1,200	
Media Backplane	7869	1	140	1	140	
Power Midplane	7870	1	200	1	200	
CPU Regulator	7875	1	1,100	3	3,300	
Serial Port Riser Card	7878	1	100	1	100	
System Drawer Enclosure	7879	1	350	1	350	
Service Processor Card (FSP)	7881	1	650	1	650	
AC Power Supply, 240V, 1400 W	7888	1	800	2	1,600	
Single Processor Activation For #7832	7898	1	21,800	2	43,600	6,768
Zero-priced Processor Activation Code for #7832	8454	1		2		6,768
Rack 1:Rack Model T00	7014-T00	1	2,920	1	2,920	768
1.8m Rack Trim Kit	6246	1	158	1	158	
PDU to Wall Powercord 14', 200-240V/24A,	6654	1	240	1	240	
HMC 1:7310-C04 Desktop Hardw.Mgmt.Console	7310-C04	1	1,830	1	1,830	1,344
IBM ThinkVision C170 17-inch Color Monitor	3631	1	250	1	250	
Power Cord (6-foot), To Wall (125V, 15A),	6470	1	14	2	28	
Ethernet Cable, 6M, Hardware Management	7801	1	12	1	12	
Quiet Touch Keyboard - USB, Business Black,	8800	1	83	1	83	
Mouse - Business Black with Keyboard Attachment Cable	8841	1	62	1	62	
			Subtotal		514,449	30,474
Storage						
DS4000 EXP710 Storage Expansion	1740-710	1	6,000	26	156,000	
Short Wave SFP	2210	1	499	109	54,391	
2Gb FC, 36.4GB/15K Drive	5212	1	1,115	338	376,870	
DS4500 Storage Server	1742-90U	1	49,900	3	149,700	
Fiber Cable 1m	5601	1	79	52	4,108	
Fiber Cable 5m	5605	1	129	6	774	
2Gb Mini HUB	3507	1	899	4	3,596	
DS4500 AIX Host Kit	7111	1	4,000	1	4,000	

	IBM eServer p5 570 Model 9117-570			TPC-C Rev. 5.4		
				Report Date: October 13, 2005		
1740-710 24x7x4 WSU	6942-25B	1	760	26	19,760	
1742-90U 24x7x4 WSU	6942-25B	1	1,087	3	3,261	
			Subtotal	749,439	23,021	
Server Software						
AIX 5.3 (media only)	5962-A5L	1	50	1	50	
AIX Software	5765-AIX	1	1,225	4	4,900	
AIX Software Maintenance (3Y)	5773-SM3	1	1,958	4	7,832	
AIX Software Maintenance 24x7 Upgrade (3Y)	0468	1	496	4	1,984	
HMC Software Support 3 Yr. Support per Processor 0569	5773-RS3	1	675	1	675	
HMC Software Support 24x7x4 / 3 Year	570	1	236	1	236	
C for AIX user Lic+SW maint 12 MO	D5A1DLL	1	515	1	515	
C for AIX user annual SW maint renewal	E1A1FLL	1	103	2	206	
Oracle Database 10g Enterprise Edition, Per Processor for 3 years		2	20,000	3‡	60,000	
Oracle Database Server Support Package for 3 years		2	6,000	1	6,000	
Oracle E-Business Discount (license & support)		2		1	-6,600	
			Subtotal	58,865	16,933	
Client Hardware and Software						
x336 3.2GHz 800MHz Xeon EM64T Express Model	8837E3U	1	3,525	8	28,200	
Express IBM 36.4GB 15k U320 SCSI HS Option	30R5096	1	275	8	2,200	
NetBay42 Standard Rack	9307-4SX	1	1,489	2	2,978	
IBM Sleek 2-Button Mouse - (PS/2)	90P0740	1	15	1	15	
Preferred Pro Full Size PS/2 Keyboard	25R6968	1	29	1	29	
IBM C170 17" CRT Monitor	90P0726	1	175	1	175	
			Subtotal	33,597	4,890	
Third Party Hardware/Software						
Visual C++ Standard Edition	254-00170	3	109	1	109	
Microsoft Windows 2000 Server	C11-00821	3	738	6	4,428	
Microsoft Problem Resolution Services		3	245	1	245	
BELKIN 16PT GIGABIT SWITCH	637031	4	91	3	272	
			Subtotal	4,809	245	
			Total	1,361,159	75,563	
Discounts are based on US list prices for similar quantities &		IBM Total System Discounts			-636,732	
configurations including pre-payment for maintenance			Three-Year Cost of Ownership		799,990	
Audited by: Francois Raab, Info Sizing (www.infosizing.com)					tpmC	203,439.87
Pricing Sources: 1 IBM, 3 Microsoft, 4 CDW.com					\$/tpmC	3.93
2 Oracle: MaryBeth Pierantoni, mary.beth.pierantoni@oracle.com (916-315-5081)						
‡ 3 = 0.75 * 4 Explanation: For the purposes of counting the number of processors which require licensing, a multicore chip with "n" cores shall be determined by multiplying "n" cores by a factor of .75						
Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you						

Numerical Quantities Summary for the IBM eServer p5 570 Model 9117-570

MQTH, computed Maximum Qualified Throughput: 203,439.87 tpmC

<u>Response Times (in seconds)</u>	<u>90th %</u>	<u>Average</u>	<u>Maximum</u>
New Order	0.98	0.42	1.71
Payment	1.00	0.43	2.20
Order-Status	1.00	0.43	1.60
Delivery (interactive)	0.15	0.15	0.65
Delivery (deferred)	0.05	0.03	0.05
Stock-Level	1.00	0.43	1.55
Menu	0.15	0.12	0.84

Response time delay added for emulated components 0.1 seconds

<u>Transaction Mix, in percent of total transactions</u>	<u>Percent</u>
New Order	44.91%
Payment	43.01%
Order-Status	4.02%
Delivery	4.01%
Stock-Level	4.01%

<u>Keying/Think Times (in seconds)</u>	<u>Min.</u>	<u>Average</u>	<u>Max.</u>
New Order	18.00/0.01	18.01/12.02	18.04/120.21
Payment	3.00/0.01	3.01/12.02	3.04/120.20
Order-Status	2.00/0.01	2.01/10.01	2.04/100.10
Delivery	2.00/0.01	2.01/5.02	2.04/50.20
Stock-Level	2.00/0.01	2.01/5.02	2.04/50.20

Test Duration

Ramp-up Time	1 hour 10 minutes 19 Seconds
Measurement interval	2 hours 00 minutes
Transactions during measurement interval (all types)	54,355,988
Ramp-down time	20 minutes

Checkpoints

Number of checkpoints	4
Checkpoint interval	29 minutes 4 sec

Table of Content

Preface	10
0 General Items	11
0.1. Application Code Disclosure	11
0.2. Benchmark Sponsor	11
0.3. Parameter Settings	11
0.4. Configuration Diagrams	11
1 Clause 1: Logical Data Base Design Related Items	13
1.1. Table Definitions	13
1.2. Database Organization	13
1.3. Insert and/or Delete Operations	13
1.4. Horizontal or Vertical Partitioning	13
2 Clause 2: Transaction and Terminal Profiles Related Items	14
2.1. Verification for the Random Number Generator	14
2.2. Input/Output Screens	14
2.3. Priced Terminal Features	14
2.4. Presentation Managers	14
2.5. Home and Remote Order-lines	14
2.6. New-Order Rollback Transactions	14
2.7. Number of Items per Order	15
2.8. Home and Remote Payment Transactions	15
2.9. Non-Primary Key Transactions	15
2.10. Skipped Delivery Transactions	15
2.11. Mix of Transaction Types	15
2.12. Queuing Mechanism of Delivery	16
3 Clause 3: Transaction and System Properties	17
3.1. Atomicity Requirements	17
3.2. Consistency Requirements	17
3.3. Isolation Requirements	18
3.4. Durability Requirements	18
4 Clause 4: Scaling and Data Base Population Related Items	20
4.1. Cardinality of Tables	20
4.2. Distribution of Tables and Logs	20
4.3. Data Base Model Implemented	21
4.4. Partitions/Replications Mapping	21
4.5. 60 Day Space Calculation	22
5 Clause 5: Performance Metrics and Response Time Related Items	23
5.1. Response Times	23
5.2. Keying and Think Times	23
5.3. Response Time Frequency Distribution	24
5.4. Performance Curve for Response Time versus Throughput	27
5.5. Think Time Frequency Distribution	27
5.6. Throughput versus Elapsed Time	28
5.7. Steady State Determination	28
5.8. Work Performed During Steady State	28
5.9. Measurement Interval	30
6 Clause 6: SUT, Driver, and Communication Definition Related Items	31
6.1. RTE Availability	31
6.2. Measurement Interval	31
6.3. Functionality and Performance of Emulated Components	31
6.4. Network Bandwidth	31
6.5. Operator Intervention	31
7 Clause 7: Pricing Related Items	32
7.1. Hardware and Programs Used	32
7.2. Three Year Cost of System Configuration	32
7.3. Availability Dates	32

7.4.	Statement of tpmC and Price/Performance.....	32
8	Clause 9: Audit Related Items.....	33
Appendix - A:	Client Server Code	38
A.1	Client/Terminal Handler Code	38
A.2	Transaction Code.....	90
A.3	TM Code.....	124
Appendix - B:	Tunable Parameters.....	176
B.1	Database Parameters.....	176
B.2	Transaction Monitor Parameters.....	176
B.3	AIX Parameters	179
Appendix - C:	Database Setup Code	181
C.1	Database Creation Scripts.....	181
C.2	SQL Creation Scripts.....	185
C.3	Data Generation Code	190
Appendix - D:	RTE Scripts.....	211
D.1	RTE Parameters.....	211
D.2	RTE Scripts	213
Appendix - E:	Third Party Pricing.....	227

Abstract

This report documents the full disclosure information required by the TPC Benchmark™ C Standard Specification Revision 5.4 dated April 2004, for measurements on the IBM eServer p5 570 Model 9117-570. The software used on the IBM eServer p5 570 Model 9117-570 includes AIX 5L Version 5.3 operating system, Oracle 10g database manager. Microsoft COM+ is used as transaction manager.

IBM eServer p5 570 Model 9117-570

Company Name	System Name	Data Base Software	Operating System Software
IBM Corporation	IBM eServer p5 570 Model 9117-570	Oracle Database 10g Enterprise Edition	AIX 5L Version 5.3

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
\$799,990 USD	203,439.87	\$3.93 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 5.4 in April 2004.

This is the full disclosure report for benchmark testing of the IBM eServer p5 570 Model 9117-570 and Oracle 10g 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 eServer pSeries application code for the five TPC Benchmark™ C transactions. Appendix D contains the terminal functions and layouts.

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.*

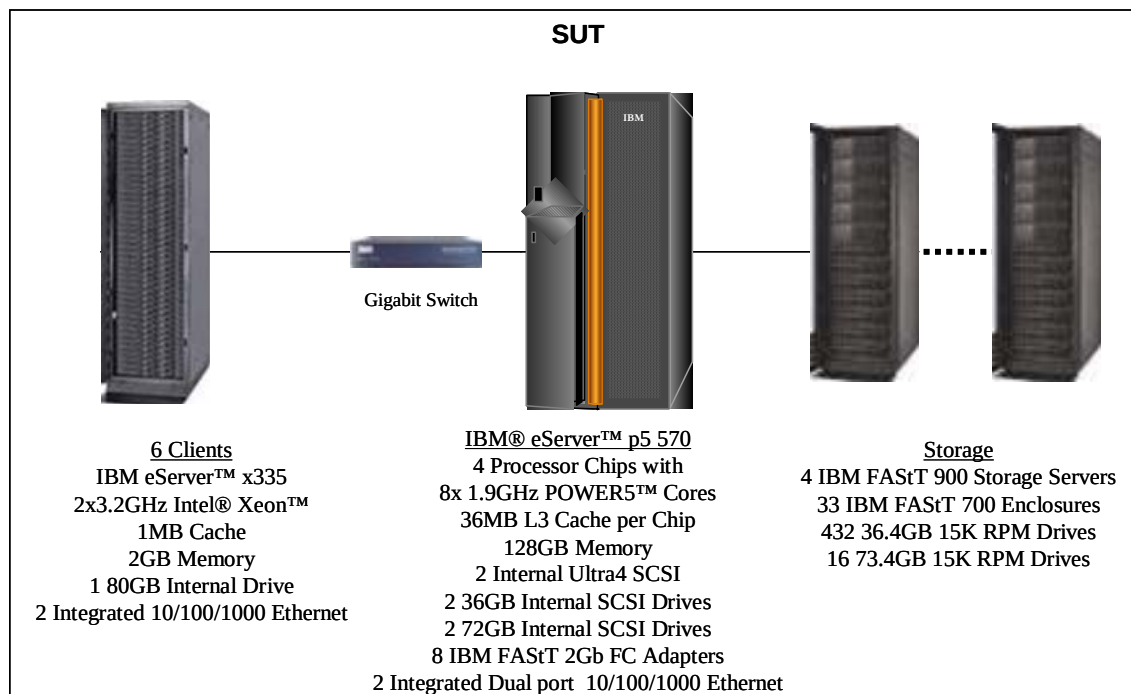
Appendix B contains the system, data base, and application parameters changed from their default values used in these TPC Benchmark™ C tests.

0.4. Configuration Diagrams

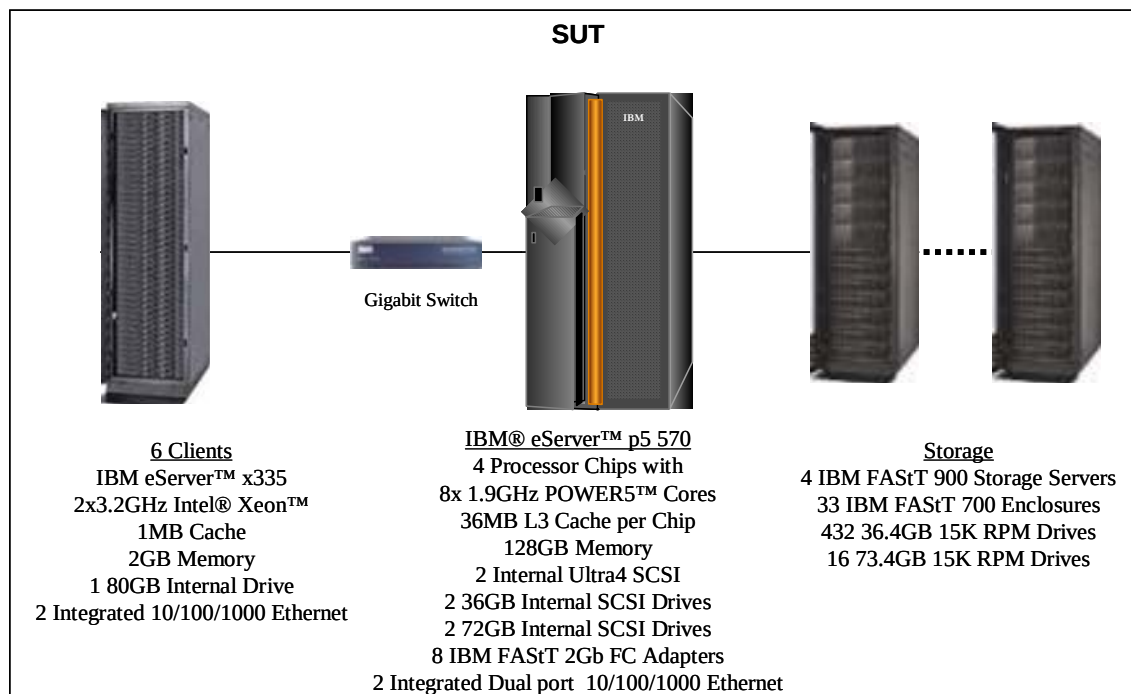
Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:

- *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 eServer p5 570 Model 9117-570 Benchmark Configuration



IBM eServer p5 570 Model 9117-570 Priced Configuration



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 Oracle Database 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 Oracle 10g and priced as static space.

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.

Partitioning was not used for any of the measurement reported in this full disclosure.

2 Clause 2: Transaction and 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 `srandom()`, `getpid()` and `gettimeofday()` functions are used to produce unique random seeds for each driver. The drivers use these seeds to seed the `srand()`, `srandom()` and `srand48()` functions. Random numbers are produced using wrappers around the standard system random number generators.

The negative exponential distribution uses the following function to generate the distribution. This function has the property of producing a negative exponential curve with a specified average and a maximum value 4 times the average.

```
const double RANDOM_4_Z = 0.89837799236185
const double RANDOM_4_K = 0.97249842407114
double neg_exp_4(double average {
    return - average * (1/RANDOM_4_Z * log (1 - RANDOM_4_K * drand48()))};
}
```

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 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 eServer x336, 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 eServer p5 570 Model 9117-570
Percentage of Home order lines	99.0%
Percentage of Remote order lines	1.00%
Percentage of Rolled Back Transactions	1.00%
Average Number of Items per order	9.99
Payment	
Percentage of Home transactions	85.01%
Percentage of Remote transactions	14.99%
Non-Primary Key Access	
Percentage of Payment using C_LAST	59.99%
Percentage of Order-Status using C_LAST	60.06%
Delivery	
Delivery transactions skipped	0
Transaction Mix	
New-Order	44.91%
Payment	43.01%
Order-Status	4.02%
Delivery	4.02%
Stock-Level	4.01%

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 scripts 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:

$$d_next_o_id - 1 = \max(o_id) = \max(no_o_id)$$

$$\text{where } (d_w_id = o_w_id = no_w_id) \text{ and } (d_id = o_d_id = no_d_id)$$

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:

$$\max(no_o_id) - \min(no_o_id) + 1 = \text{number of rows in New Order for the Warehouse/District}$$

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:

$$\text{sum}(o_ol_cnt) = \text{number of rows in the Order Line table for the Warehouse/District}$$

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

3.4.1. Permanent Unrecoverable Failure of any Single Durable Medium

Permanent irrecoverable failure of any single durable medium containing TPC-C data base tables or recovery log data.

Failure of Log Disk and Log Cache:

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

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. One of the disks containing the transaction log was removed. Since the disk was RAID-5 (mirrored & striped), Oracle 10g Server continued to process the transactions successfully.
4. The test continued for at least another 5 minutes.
5. Since write cache mirroring was enabled for the log device, one of the Fibre Channel controllers, which holds one copy of the mirrored cache, was removed. There was a brief pause in I/O while the failover to the remaining log controller occurred. The controller detected a mirror-out-of-sync condition and deactivated write-back cache.
6. The run continued without write-back cache.
7. The system was subsequently shut down.

8. The disk from step 3 was replaced.
9. The system was powered back on and Oracle 10g Server was allowed to recover.
10. 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.

Failure of Durable Medium Containing TPC-C Database Tables:

The following steps were successfully performed to demonstrate Durability against the failure of a disk unit with database tables:

1. The contents of the database were backed up in full.
2. 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.
3. A scaled-down test was started with about 20% of the full load.
4. A disk containing the TPCC table was removed causing Oracle 10g Server to report numerous errors when attempting to access that device
5. The disk was powered back on and the full database was restored from the backup copy in step 1.
6. Oracle 10g Server was restarted and the transactions in the log were applied to the database.
7. Step 2 was performed returning 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.
8. Consistency condition 3 was verified.

Instantaneous Interruption and Memory Failure:

The following steps were conducted on a fully scaled database:

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 powered off, which removed power from all system components, including memory.
4. The system was powered back on and Oracle 10g Server recovered.
5. Step 1 was performed returning 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
6. 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 17,600 warehouses, the number of active warehouses during the benchmark was 16,256.

Table Name	Number of Rows
Warehouse	17,600
District	176,000
Customer	528,000,000
History	528,000,000
Order	528,000,000
New Order	158,400,000
Order Line	5,280,144,056
Stock	1,760,000,000
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 described for the tested and priced systems.

The system is configured with 16 RAID0 for the tables and 2 RAID5 for the log.

For the tables, each RAID0 (hdisk) is configured with 20x 36.4GB disks with a capacity of 668GB per RAID. There are a total of 172 Logical Volumes (LV) with 6 pairs of storage adapters and 1 volume group. Each adapter has 4 hdisks and each LV is spread across all the disks.

For the log, two RAID5 arrays are configured with 9 x 36.4GB disk each with a total capacity of 534.25GB. There are two Logical Volume (LV) with 1 storage controller. The controller has 2 logical disk (hdisks) and the log LV is striped across the 2 hdisks.

Disk	volume
Hdisk3- hdisk10, hdisk13- 20	tpc_ware_0_0 tpc_cust_0_0 tpc_cust_0_1 tpc_cust_0_2 tpc_cust_0_3 tpc_cust_0_4 tpc_cust_0_5 tpc_cust_0_6 tpc_cust_0_7 tpc_cust_0_8 tpc_cust_0_9 tpc_cust_0_10 tpc_cust_0_11 tpc_cust_0_12 tpc_cust_0_13 tpc_cust_0_14 tpc_cust_0_15 tpc_cust_0_16 tpc_cust_0_17 tpc_cust_0_18 tpc_cust_0_19 tpc_cust_0_21 tpc_cust_0_20 tpc_cust_0_22 tpc_cust_0_23 tpc_cust_0_24 tpc_cust_0_25 tpc_cust_0_26 tpc_cust_0_27 tpc_cust_0_28 tpc_cust_0_29 tpc_cust_0_30 tpc_cust_0_31 tpc_cust_0_32 tpc_cust_0_33 tpc_cust_0_34 tpc_cust_0_36 tpc_cust_0_35 tpc_cust_0_37 tpc_cust_0_38 tpc_cust_0_39 tpc_cust_0_40 tpc_cust_0_41 tpc_cust_0_42 tpc_cust_0_43 tpc_cust_0_44 tpc_cust_0_45 tpc_cust_0_46 tpc_cust_0_47 tpc_cust_0_48 tpc_cust_0_49 tpc_cust_0_50 tpc_cust_0_51 tpc_cust_0_52 tpc_cust_0_53 tpc_cust_0_54 tpc_cust_0_55 tpc_cust_0_56 tpc_cust_0_57 tpc_cust_0_58 tpc_dist_0_0 tpc_hist_0_0 tpc_hist_0_1 tpc_hist_0_2 tpc_hist_0_3 tpc_hist_0_4 tpc_hist_0_5 tpc_stok_0_0 tpc_stok_0_1 tpc_stok_0_2 tpc_stok_0_3 tpc_stok_0_4 tpc_stok_0_5 tpc_stok_0_6 tpc_stok_0_7 tpc_stok_0_8 tpc_stok_0_9 tpc_stok_0_10 tpc_stok_0_11 tpc_stok_0_12 tpc_stok_0_13 tpc_stok_0_14

Disk	volume
	tpc_stok_0_15 tpc_stok_0_16 tpc_stok_0_17 tpc_stok_0_18 tpc_stok_0_19 tpc_stok_0_20 tpc_stok_0_21 tpc_stok_0_22 tpc_stok_0_23 tpc_stok_0_24 tpc_stok_0_25 tpc_stok_0_26 tpc_stok_0_27 tpc_stok_0_28 tpc_stok_0_29 tpc_stok_0_30 tpc_stok_0_31 tpc_stok_0_32 tpc_stok_0_33 tpc_stok_0_34 tpc_stok_0_35 tpc_stok_0_36 tpc_stok_0_37 tpc_stok_0_38 tpc_stok_0_39 tpc_stok_0_40 tpc_stok_0_41 tpc_stok_0_42 tpc_stok_0_43 tpc_stok_0_44 tpc_stok_0_45 tpc_stok_0_46 tpc_stok_0_47 tpc_stok_0_48 tpc_stok_0_49 tpc_stok_0_50 tpc_stok_0_51 tpc_stok_0_52 tpc_stok_0_53 tpc_stok_0_54 tpc_stok_0_55 tpc_stok_0_56 tpc_stok_0_57 tpc_stok_0_58 tpc_stok_0_59 tpc_stok_0_60 tpc_stok_0_61 tpc_stok_0_62 tpc_stok_0_63 tpc_stok_0_64 tpc_stok_0_65 tpc_item_0_0 tpc_ordr_0_0 tpc_ordr_0_1 tpc_ordr_0_2 tpc_ordr_0_3 tpc_ordr_0_4 tpc_ordr_0_5 tpc_ordr_0_6 tpc_ordr_0_7 tpc_ordr_0_8 tpc_ordr_0_9 tpc_nord_0_0 tpc_iware_0_0 tpc_icust1_0_0 tpc_icust1_0_1 tpc_icust2_0_0 tpc_icust2_0_1 tpc_icust2_0_2 tpc_icust2_0_3 tpc_idist_0_0 tpc_idist_0_1 tpc_istok_0_0 tpc_istok_0_1 tpc_iitem_0_0 tpc_iordr2_0_0 tpc_iordr2_0_1 tpc_iordr2_0_2 tpc_iordr2_0_3 tpc_tpccaux tpc_control_001 tpc_control_002 tpc_sp_0 tpc_log_1_1 tpc_log_1_2 tpc_system_1 tpc_roll1 tpc_dist_0_1 tpc_idist_0_2 tpc_stok_0_66 tpc_system_2

Table 4-2: IBM eServer p5 570 Model 9117-570 Data Distribution

4.3. Data Base Model Implemented

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

The relational database manager used for this test was Oracle Database 10g Enterprise Edition.

4.4. Partitions/Replications Mapping

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

IBM did not implement horizontal or vertical partitioning for this TPC-C test.

4.5. 60 Day Space Calculation

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.

60 Day Space Calculations							
TPM	203439						
Warehouses	17600						
SEGMENT	TYPE	BLOCKS	SIZE	FIVE PCT	DAILY GROW	TOTAL (MB)	
CUSTCLUSTER	CLUSTER	184997452	2048	9249873	0	379389.31	
DB_STAT	SYS	1048576	2048	0	0	2048.00	
DISTCLUSTER	CLUSTER	183552	2048	9178	0	376.43	
HIST	TABLE	16188795	2048	0	2994029	37466.45	
ICUST1	INDEX	992560	16384	49628	0	16284.19	
ICUST2	INDEX	14414392	2048	720720	0	29560.76	
IDIST	INDEX	89024	2048	4451	0	182.57	
IITEM	INDEX	22528	2048	1126	0	46.20	
IORDR2	INDEX	10909520	2048	545476	0	22373.04	
ISTOCK	INDEX	2642328	16384	132116	0	43350.69	
ITEMCLUSTER	CLUSTER	8446	2048	422	0	17.32	
IWARE	INDEX	23024	2048	1151	0	47.22	
NORDCLUSTER_QUEUE	CLUSTER	2139053	2048	106953	0	4386.73	
ORDRCLUSTER_QUEUE	CLUSTER	26775340	16384	0	4951953	495738.95	
STOKCLUSTER	CLUSTER	251697303	2048	12584865	0	516176.11	
SYSAUX	SYS	61440	2048	0	0	120.00	
SYSTEM	SYS	204800	2048	0	0	400.00	
SYS_IQ0000011024\$\$	SYS	109735	16384	5487	0	1800.34	
SYS_IQ0000011070\$\$	SYS	305579	2048	15279	0	626.68	
WARECLUSTER	CLUSTER	18816	2048	941	0	38.59	
Total		512,832,263		23,427,666	7,945,982	1,550,429.57	
Dynamic space	449,983 Initial MB for (History+Orde						
Static space	1,017,224 Initial blocks +5% -Dynamic						
Daily growth	83,222						
Daily spread	0 Oracle may be configured such that daily spread is 0						
60-day space (MB)	6,010,543						
60-day (GB)	5,869.67						
		new_order		38457334			
Log block size	512 Redo blocks written			396865974			
Log blocks/tpmC	10.32 Number of log blocks used in one tpmC						
8-hour log (GB)	480.52						
Disk	Disk	SUT					
	Formatted						
Type	Capacity	No of arrays	Capacity(GB)	Data	10688.00		
				Log	534.25		
				OS	67.75		
Data RAID0(20-36GB)	684032	16	10688.00	Total Spac	11290.00		
Log RAID5(9-36GB)	273536	2	534.25				

Table 4-3: Space Calculation

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%	0.98	1.00	1.00	0.15/0.05	1.00	0.15
Average	0.42	0.43	0.43	0.15/0.03	0.43	0.12
Maximum	1.71	2.20	1.6	0.65/0.05	1.55	0.84
Think Times						
Minimum	0.01	0.01	0.01	0.01	0.01	N/A
Average	12.02	12.02	10.01	5.02	5.02	N/A
Maximum	120.22	120.20	100.10	50.20	50.21	N/A
Keying Times						
Minimum	18.00	3.00	2.00	2.00	2.00	N/A
Average	18.01	3.01	2.01	2.01	2.01	N/A
Maximum	18.05	3.05	2.05	2.06	2.05	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.

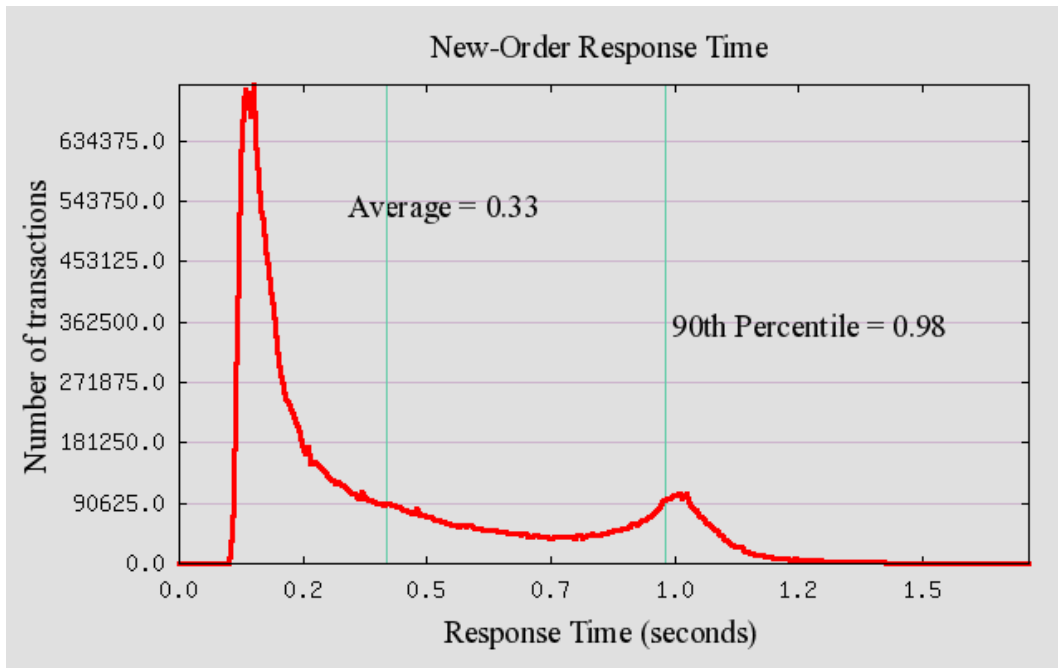


Figure 5-1: New-Order Response Time Distribution

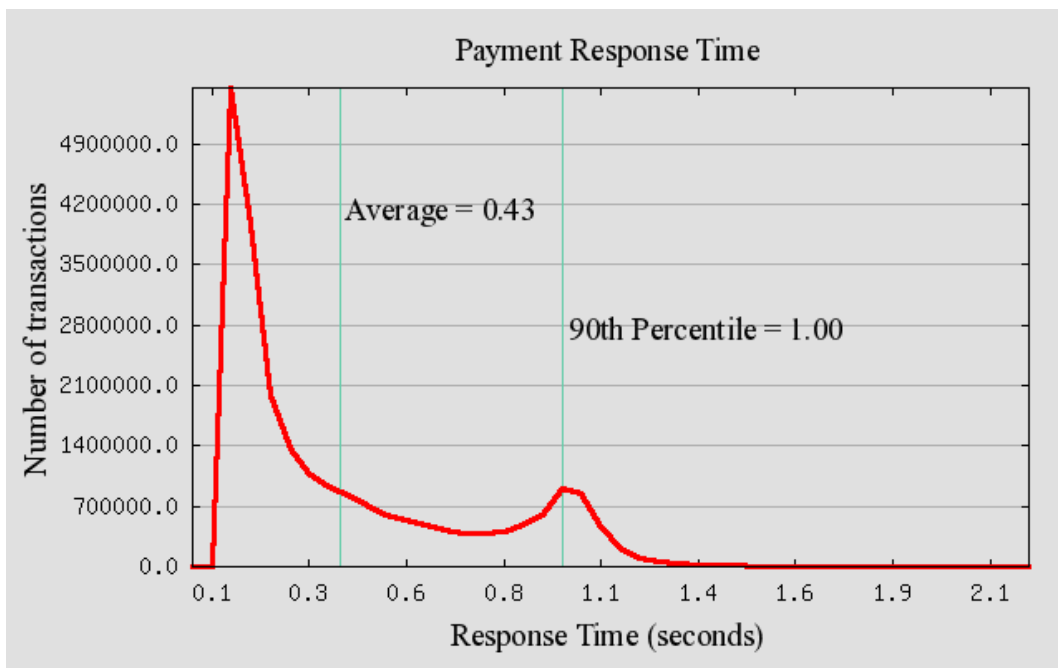


Figure 5-2: Payment Response Time Distribution

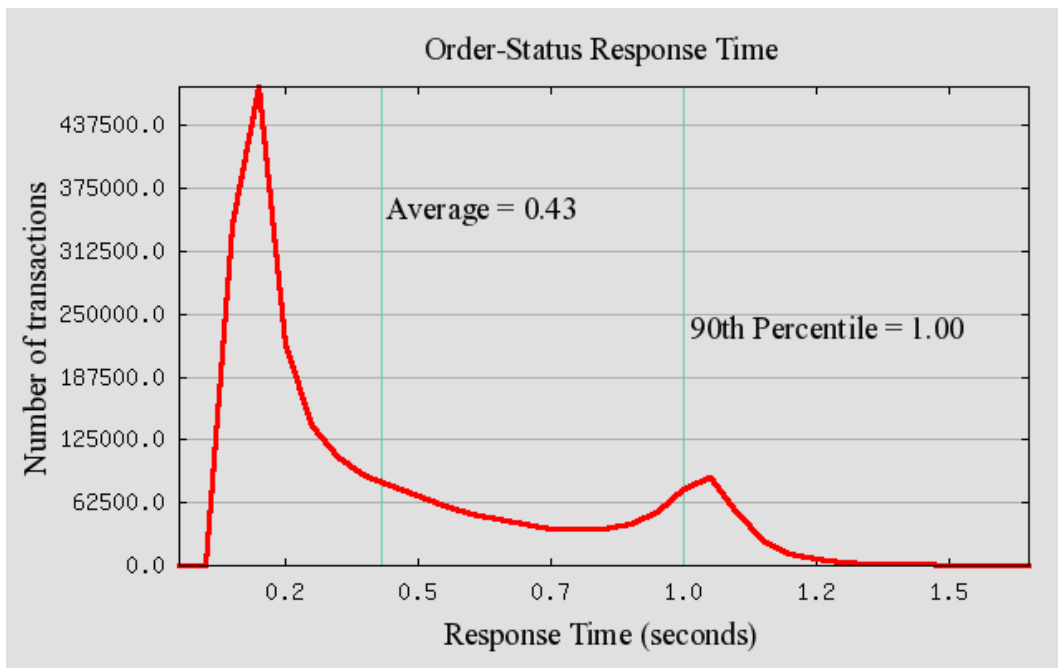


Figure 5-3: Order-Status Response Time Distribution

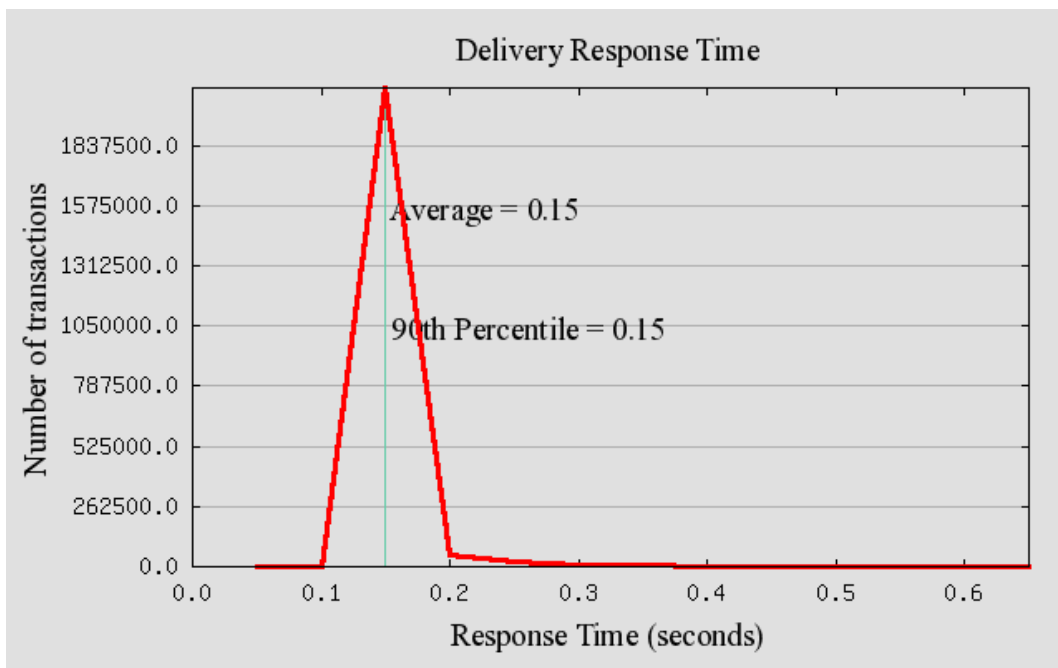


Figure 5-4: Delivery (Interactive) Response Time Distribution

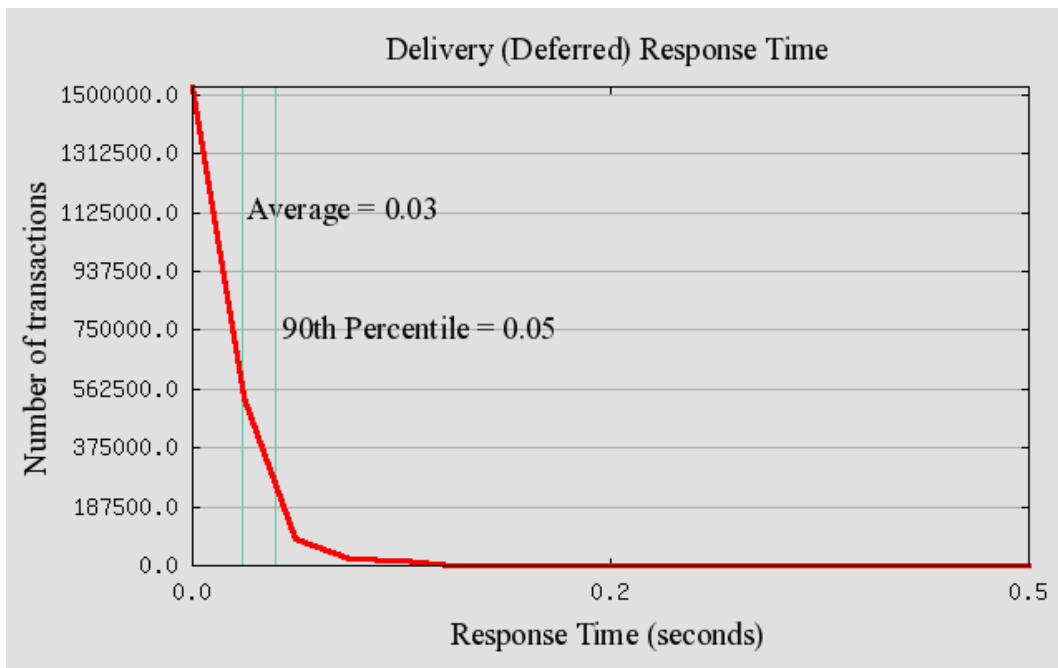


Figure 5-5: Delivery (Deferred) Response Time Distributio

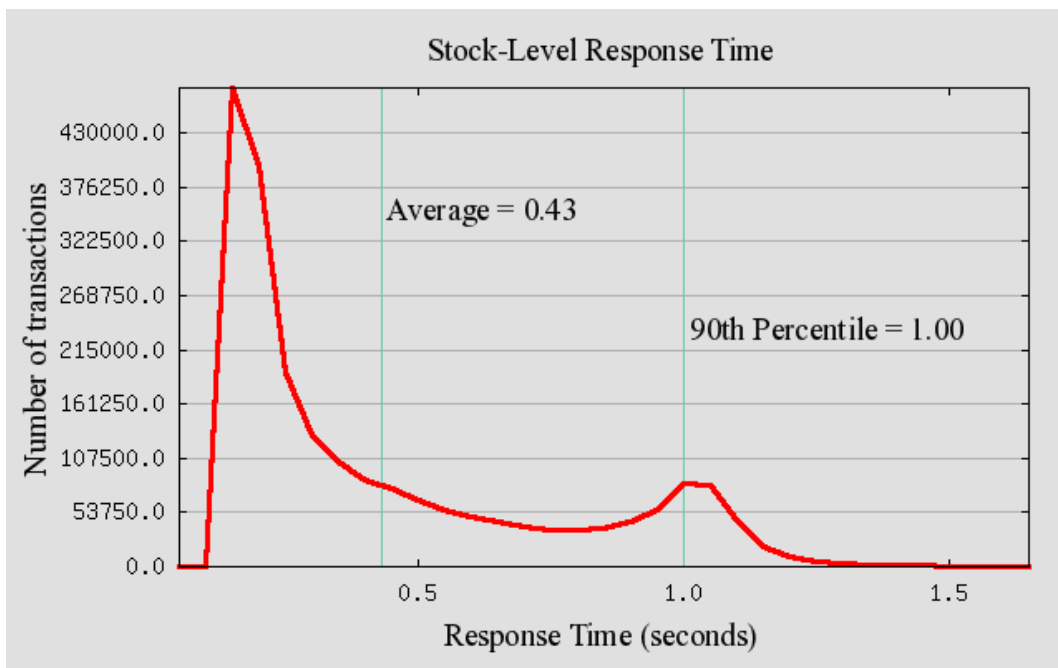


Figure 5-6: Stock Level Response Time Distribution

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.

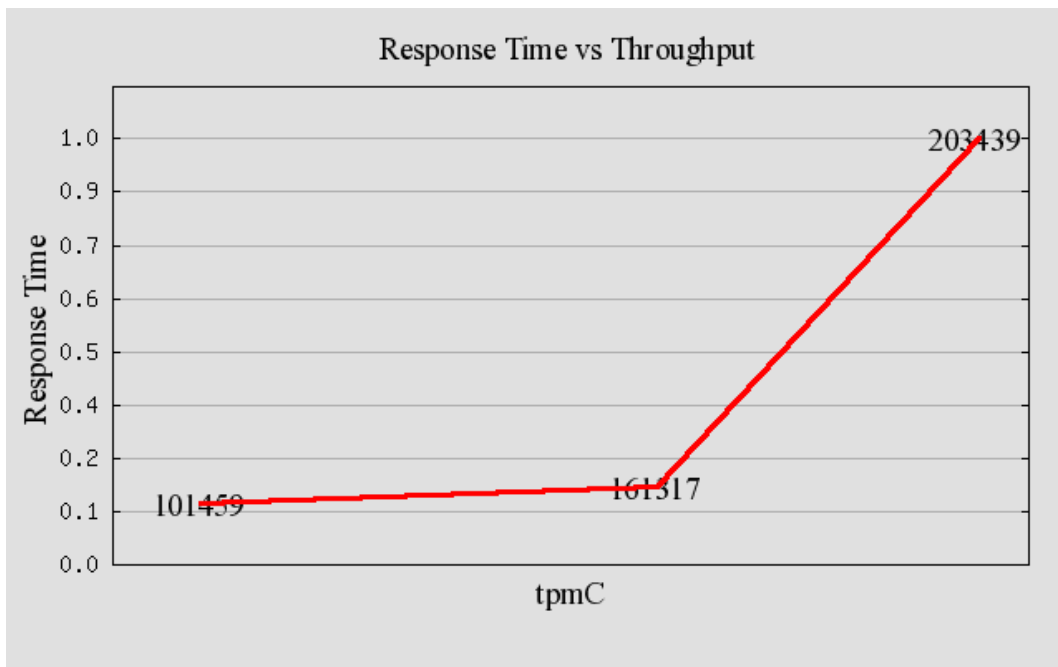


Figure 5-7: New-Order Response Time vs. Throughput

5.5. Think Time Frequency Distribution

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

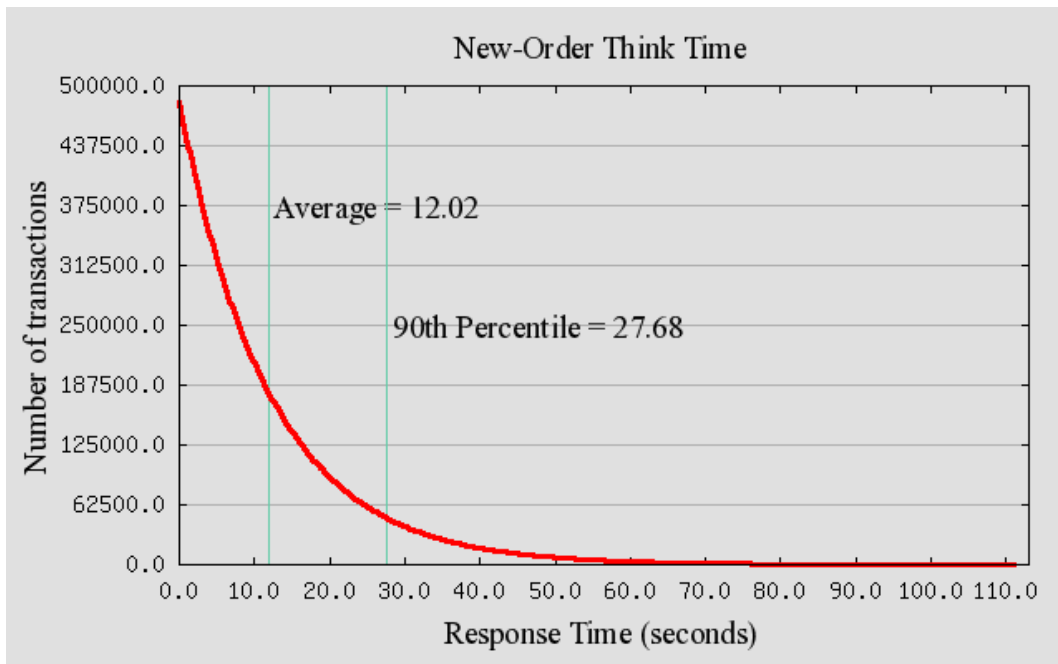


Figure 5-8: New-Order Think Time Distribution

5.6. Throughput versus Elapsed Time

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

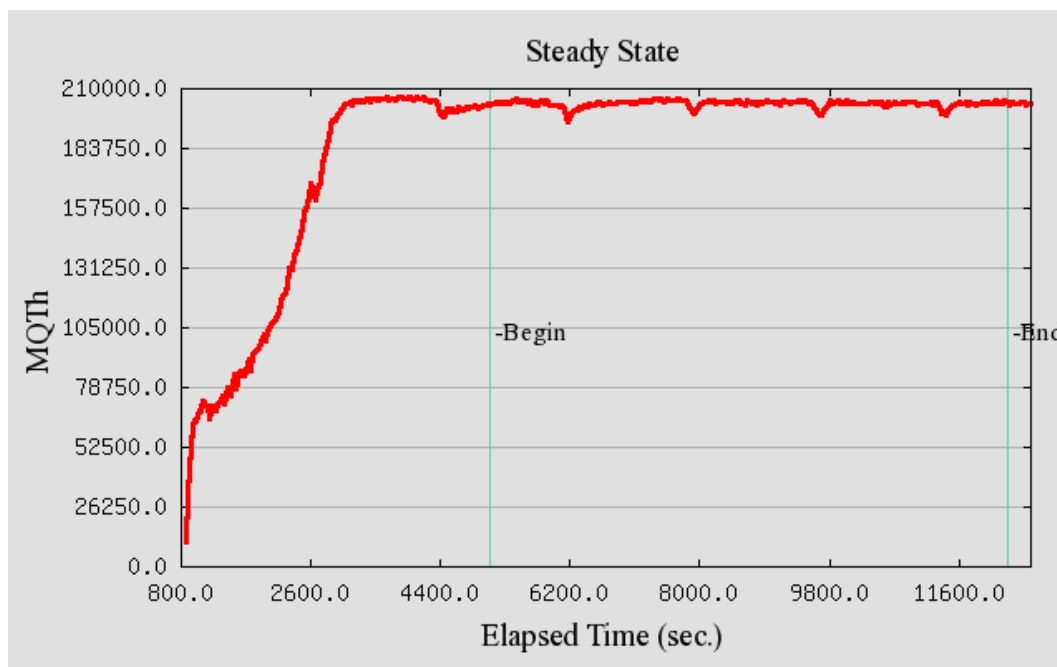


Figure 5-9: New-Order Throughput vs. Elapsed Time

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 5-9 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 measurement interval was used to guaranty that all work normally performed during an 8-hour sustained test are included in the reported throughput.

5.8.1. Transaction Flow

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

Microsoft COM+ for Windows 2000 Server was used as a Transaction Monitor(TM). The transaction framework was divided into three programs. The front end program, Microsoft Internet Information Server (IIS), handled all HTML requests through an ISAPI module, a back end database client program (COM+), which connected to the database, and a database server program which handled all database operations at the SUT (Oracle10g). Both the front and back end programs ran on the client systems. The front end program communicates with database client programs through COM+ RPC calls. IIS ISAPI code communicates with the clients through HTML over Ethernet. The database client program communicates with the Server system over Ethernet using SQL*Net calls. Besides calling initialization code during startup, all other functions are transparent to the application code. IIS and COM+ routes the transactions and balances the load according to the options defined in the configuration file (Windows Registry) in appendix B.2, IIS ISAPI code is

capable based on this configuration file to direct transactions to COM+ pools. Delivery Transactions are Asynchronous and are handled in the IIS ISAPI code by implementing multiple configurable threads to process requests from a common queue. The transaction flow is described below.

- COM+ is configured with 1 pool having 19 objects each with a connection to the database..
- ISAPI is configured with 2 threads to handle delivery threads.
- A total of 20500 browser entries are set up in ISAPI.
- IIS is configured with 800 threads to handle the communication with the users.
- When the users login to the webserver the entire framework is established, including the connections to the database and the common Browser Array in ISAPI.
- Number of warehouses per client is configured as 2040
- When browsers are started, each emulated user connects to the IIS webserver, which has been configured to have all html pages handled by ISAPI code disclosed in Appendix A.
- The TPC-C user chooses the transaction type and proceeds to fill the screen fields required for the transaction.
- The ISAPI code accepts all values entered by the user and transmits those values to one of the TPC-C back end programs. The transaction is performed through a COM+ RPC. There is an interface for each TPC-C transaction type and each TPC-C back end program exports these interfaces. COM+ transparently routes the RPC to one of the servers exporting the corresponding handle.
- The delivery transactions are executed by placing the request on a queue in ISAPI.
- A TPC-C back-end server program receives a RPC and proceeds to execute all database operations related to the request. All information entered on the users emulated browser is contained in the RPC.
- Once the transaction is done (committed or rolled back), the server program fills in the output parameters. The RPC is then sent back to the ISAPI code.
- When the RPC returns to the ISAPI code, the routine writes the transaction out on the users browser

5.8.2. Database Transaction

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

Using SQL*Net calls, the TPC-C back-end program interacts with Oracle 10g 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.

Oracle 10g Server proceeds to update the database as follows:

When Oracle 10g 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, Oracle 10g Server will make space by flushing some modified pages to disk. Modified pages are also written to disk when a checkpoint occurs. Before a change is made to the database, it is first recorded in the transaction log. This ensures that the database can be recovered completely in the event of a failure. Using the transaction log, transactions that started but did not complete prior to a failure can be undone, and transactions recorded as complete in the transaction log but not yet written to disk can be redone.

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 30 minutes. Two checkpoints occur during the rampup period, and four others occurring during the measurement interval. The incremental checkpoint duration during the measurement is 29 minutes and 4 seconds.

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 measurement interval was used.

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. Appendix D contains the scripts used in the testing.

6.2. Measurement Interval

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

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

6.3. 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.

6.4. 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.

6.5. 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.

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 E. All prices are based on IBM US list prices.

Discounts are based on US list prices and for similar quantities and configurations.

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 products are generally available now.

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 eServer p5 570 Model 9117-570	203,439.87	\$799,990 USD	\$3.93 USD	October 14, 2005

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.

The auditor's attestation letter is included in this section of this report:

Benchmark Sponsor: John J. Makis
IBM eServer Performance
11501 Burnet Road
Austin, TX 78758

October 12, 2005

I verified the TPC Benchmark™ C performance for the following Client/Server configuration:

Platform: **IBM eServer p5 570 Model 9117-570c/s**
Operating system: **AIX 5L V5.3**
Database Manager: **Oracle Database 10g Enterprise Edition**
Transaction Manager: **Microsoft COM+**

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: IBM eServer p5 570 Model 9117-570				
4 x POWER5 (1.9GHz)	128 GB Main (36MB L3 Cache/DCM)	338 x 36.4 GB ext. 2 x 36.4 GB int.	0.98 Seconds	203,439.87
Clients: Eight (8) IBM @server xSeries 335 Model 8676-G2X (Specification for each)				
2 x Intel Xeon (3.2 GHz)	2 GB Main (1 MB L3 Cache per processor)	1 x 80 GB	n/a	n/a

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

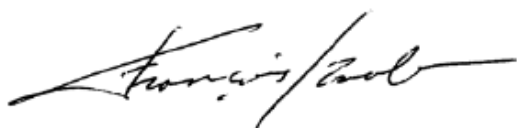
- The transactions were correctly implemented
- The database records were the proper size.
- The database was properly scaled and populated.

- The required ACID properties were met
- The transactions were correctly implemented
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.
- All 90% response times were under the specified maximums
- At least 90% of all delivery transactions met the 80 Second completion time limit
- The reported measurement interval was 120 minutes (7200 seconds)
- The reported measurement interval was representative of steady state conditions
- Four checkpoints were taken during the reported measurement interval
- The 60 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

None.

Respectfully Yours,



François Raab, President



Bradley J. Askins, Auditor

Appendix - A: Client Server Code

A.1 Client/Terminal Handler Code

tpccIsapi.def

; tpccIsapi.def : declares the module parameters for the DLL.

LIBRARY "tpccIsapi"

EXPORTS

HttpExtensionProc
GetExtensionVersion
TerminateExtension

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
#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 NEGATIVE_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_NUMSERVERS_NOT_IN_REG -56
#define ERR_NUMWAREHOUSES_NOT_IN_REG -57

#define ERR_INVALID_USERNAME -70
#define ERR_INVALID_PASSWORD -71
#define ERR_INVALID_DB_NAME -72
#define ERR_INVALID_REGISTRY_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_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

////////////////////
// Registry Definitions
////////////////////
#define REGISTRY_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 {
    struct in_neword_struct in_nord;
    struct out_neword_struct 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_dlv;

```

```

    struct out_delivery_struct out_dlv;
};

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 {
        double s_I_PRICE;
        double 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;
        double 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 appended
//      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 EPOCH 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 remaining 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    negativeFlag= 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 negative flag
                negativeFlag = 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 negative
    if(negativeFlag)
        *number = *number * (-1);
}

```

```

        return true;
    }

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

        bool    negativeFlag= 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 negative flag
                    negativeFlag = 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 negative
if(negativeFlag)
    *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    negativeFlag= false;
bool    moneyFlag    = true;

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

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

// handle negative value
if(value < 0)
{
    negativeFlag = 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 negative indicator
                if(negativeFlag)
                {
                    negativeFlag = false;
                    buffer[--index] = NEGATIVE_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 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;

    // 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 int index    = len;

int      places      = 0;

bool     negativeFlag= false;

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

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

// handle negative value
if(value < 0)
{
    negativeFlag = 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 negative indicator
                if(negativeFlag)
                {
                    negativeFlag = false;
                    buffer[--index] = NEGATIVE_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;
}

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

#ifdef TMPBUF
#define TMPBUF
extern char tmpbuf[128];
#endif

#define ERRORMSG(TEXT) \
    \
    EnterCriticalSection(&errorMutex); \
    \
    _strtime( tmpbuf ); \
    errorStream << debugFileName(__FILE__) \
    << "|" << tmpbuf << "|" << __LINE__ << "|" \
    << _getpid() << "|" << GetCurrentThreadId() << "|" \
    << TEXT; \
    errorStream.flush(); \
    LeaveCriticalSection(&errorMutex);

#ifdef _DEBUG

#define DEBUGMSG(TEXT) \
    \
    EnterCriticalSection(&debugMutex); \
    \
    debugStream << debugFileName(__FILE__) \
    << "|" << __TIMESTAMP__ << "|" << __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 */

```

htmlPhraser.h

```

/////////////////////////////////////////////////////////////////
// htmlPhraser.h
/////////////////////////////////////////////////////////////////
// Class to decode a html query string
/////////////////////////////////////////////////////////////////

```

```

#pragma once

#include <memory.h>

// Definitions
// Definitions
// 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
// Command Codes
// 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
// Class htmlPhraser

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

// getters
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 tpccsapi.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.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
#ifdef _DEBUG
#define ATL_CRITICAL_ISAPI_ERROR_LOGONLY
#endif

#ifdef _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

```

htmlPhraser.cpp

```

////////////////////////////////////
// htmlPhraser.cpp
////////////////////////////////////
// Class implementation of htmlPhraser.
// This class will take a query string and break it into a series
// of constituent parts
////////////////////////////////////

#include "htmlPhraser.h"

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

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

    // this initializes the query list to NULL's. This means that
    // characters being added are overwriting null characters and
    // therefore 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 character of query string

```

```

while(*queryString)
{
    // check for special case characters
    if(queryChar)
    {
        // a percentage sign would indicate a token
        if(*queryString != '%')
        {
            // a plus sign represents 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 looking 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;
}

```

////////////////////////////////////

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

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 "../tpccCom0/tpccCom0.h"
#include "../tpccCom0/tpccCom0_i.c"
#ifdef PARTITION
#include "../tpccCom1/tpccCom1.h"
#include "../tpccCom1/tpccCom1_i.c"
#include "../tpccCom2/tpccCom2.h"
#include "../tpccCom2/tpccCom2_i.c"
#include "../tpccCom3/tpccCom3.h"

```

```

#include "../tpccCom3/tpccCom3_i.c"
#include "../tpccCom4/tpccCom4.h"
#include "../tpccCom4/tpccCom4_i.c"
#include "../tpccCom5/tpccCom5.h"
#include "../tpccCom5/tpccCom5_i.c"
#include "../tpccCom6/tpccCom6.h"
#include "../tpccCom6/tpccCom6_i.c"
#include "../tpccCom7/tpccCom7.h"
#include "../tpccCom7/tpccCom7_i.c"
#include "../tpccCom8/tpccCom8.h"
#include "../tpccCom8/tpccCom8_i.c"
#include "../tpccCom9/tpccCom9.h"
#include "../tpccCom9/tpccCom9_i.c"
#include "../tpccCom10/tpccCom10.h"
#include "../tpccCom10/tpccCom10_i.c"
#include "../tpccCom11/tpccCom11.h"
#include "../tpccCom11/tpccCom11_i.c"
#include "../tpccCom12/tpccCom12.h"
#include "../tpccCom12/tpccCom12_i.c"
#include "../tpccCom13/tpccCom13.h"
#include "../tpccCom13/tpccCom13_i.c"
#include "../tpccCom14/tpccCom14.h"
#include "../tpccCom14/tpccCom14_i.c"
#include "../tpccCom15/tpccCom15.h"
#include "../tpccCom15/tpccCom15_i.c"
#include "../tpccCom16/tpccCom16.h"
#include "../tpccCom16/tpccCom16_i.c"
#include "../tpccCom17/tpccCom17.h"
#include "../tpccCom17/tpccCom17_i.c"
#include "../tpccCom18/tpccCom18.h"
#include "../tpccCom18/tpccCom18_i.c"
#include "../tpccCom19/tpccCom19.h"
#include "../tpccCom19/tpccCom19_i.c"
#include "../tpccCom20/tpccCom20.h"
#include "../tpccCom20/tpccCom20_i.c"
#include "../tpccCom21/tpccCom21.h"
#include "../tpccCom21/tpccCom21_i.c"
#include "../tpccCom22/tpccCom22.h"
#include "../tpccCom22/tpccCom22_i.c"
#include "../tpccCom23/tpccCom23.h"
#include "../tpccCom23/tpccCom23_i.c"
#include "../tpccCom24/tpccCom24.h"
#include "../tpccCom24/tpccCom24_i.c"
#include "../tpccCom25/tpccCom25.h"
#include "../tpccCom25/tpccCom25_i.c"
#include "../tpccCom26/tpccCom26.h"
#include "../tpccCom26/tpccCom26_i.c"
#include "../tpccCom27/tpccCom27.h"
#include "../tpccCom27/tpccCom27_i.c"
#include "../tpccCom28/tpccCom28.h"
#include "../tpccCom28/tpccCom28_i.c"
#include "../tpccCom29/tpccCom29.h"
#include "../tpccCom29/tpccCom29_i.c"
#include "../tpccCom30/tpccCom30.h"
#include "../tpccCom30/tpccCom30_i.c"
#include "../tpccCom31/tpccCom31.h"
#include "../tpccCom31/tpccCom31_i.c"
#include "../tpccCom32/tpccCom32.h"
#include "../tpccCom32/tpccCom32_i.c"
#include "../tpccCom33/tpccCom33.h"
#include "../tpccCom33/tpccCom33_i.c"
#include "../tpccCom34/tpccCom34.h"
#include "../tpccCom34/tpccCom34_i.c"
#include "../tpccCom35/tpccCom35.h"
#include "../tpccCom35/tpccCom35_i.c"
#include "../tpccCom36/tpccCom36.h"
#include "../tpccCom36/tpccCom36_i.c"
#include "../tpccCom37/tpccCom37.h"
#include "../tpccCom37/tpccCom37_i.c"
#include "../tpccCom38/tpccCom38.h"
#include "../tpccCom38/tpccCom38_i.c"
#include "../tpccCom39/tpccCom39.h"

```

```

#include "../tpccCom39/tpccCom39_i.c"
#include "../tpccCom40/tpccCom40.h"
#include "../tpccCom40/tpccCom40_i.c"
#include "../tpccCom41/tpccCom41.h"
#include "../tpccCom41/tpccCom41_i.c"
#include "../tpccCom42/tpccCom42.h"
#include "../tpccCom42/tpccCom42_i.c"
#include "../tpccCom43/tpccCom43.h"
#include "../tpccCom43/tpccCom43_i.c"
#include "../tpccCom44/tpccCom44.h"
#include "../tpccCom44/tpccCom44_i.c"
#include "../tpccCom45/tpccCom45.h"
#include "../tpccCom45/tpccCom45_i.c"
#include "../tpccCom46/tpccCom46.h"
#include "../tpccCom46/tpccCom46_i.c"
#include "../tpccCom47/tpccCom47.h"
#include "../tpccCom47/tpccCom47_i.c"
#include "../tpccCom48/tpccCom48.h"
#include "../tpccCom48/tpccCom48_i.c"
#include "../tpccCom49/tpccCom49.h"
#include "../tpccCom49/tpccCom49_i.c"
#include "../tpccCom50/tpccCom50.h"
#include "../tpccCom50/tpccCom50_i.c"
#endif
#include "tpccIsapi.hpp"

```

```

#include <math.h>

```

```

// 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

```

```

char tmpbuff[128];

```

```

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

```

```

int    maxDataSize;      //max struct size of all txn(s)
int    numUsers;         //number of users that client will service.
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 available slot in dlvy
                        txn queue
int    dlvyBufferThreadIndex; //thread index into dlvy txn queue
int    nullDB;           //null db on client(bypass com call).
int    trace;
int    numServers;
int    numWarehouse;
int    ratio;

```

```

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
*connectHandle);
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");

    // Initialize isapi critical section
InitializeCriticalSection(&isapiLock);

    // Initialize error log critical section
InitializeCriticalSection(&errorLock);

    // Initialize terminal critical section
InitializeCriticalSection(&termLock);

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

    // Read registry values
if(readRegistryValues() != OK)
    return(FALSE);

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

    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_struct)<<endl);
    DEBUGMSG("sizeof in_neword_struct: " <<sizeof(struct
in_neword_struct)<<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 hresults = NULL;
    HRESULT hres[50];

    try
    {
        DEBUGMSG("Got isapiLock, initializing txnHandle:
"<<DEBUGADDRESS(*txnHandle)<< endl);
        ERRORMSG("Got isapiLock, initializing txnHandle:
"<<DEBUGADDRESS(*txnHandle)<< "numServers
"<<numServers<<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.comHandle0 = NULL;
#ifdef PARTITION
(*txnHandle)->comInterface.comHandle1 = NULL;
(*txnHandle)->comInterface.comHandle3 = NULL;
(*txnHandle)->comInterface.comHandle4 = NULL;
(*txnHandle)->comInterface.comHandle5 = NULL;
(*txnHandle)->comInterface.comHandle6 = NULL;
(*txnHandle)->comInterface.comHandle7 = NULL;
(*txnHandle)->comInterface.comHandle8 = NULL;
(*txnHandle)->comInterface.comHandle9 = NULL;
(*txnHandle)->comInterface.comHandle10 = NULL;
(*txnHandle)->comInterface.comHandle11 = NULL;
(*txnHandle)->comInterface.comHandle12 = NULL;
(*txnHandle)->comInterface.comHandle13 = NULL;
(*txnHandle)->comInterface.comHandle14 = NULL;
(*txnHandle)->comInterface.comHandle15 = NULL;
(*txnHandle)->comInterface.comHandle16 = NULL;
(*txnHandle)->comInterface.comHandle17 = NULL;
(*txnHandle)->comInterface.comHandle18 = NULL;
(*txnHandle)->comInterface.comHandle19 = NULL;
(*txnHandle)->comInterface.comHandle20 = NULL;
(*txnHandle)->comInterface.comHandle21 = NULL;
(*txnHandle)->comInterface.comHandle22 = NULL;
(*txnHandle)->comInterface.comHandle23 = NULL;
(*txnHandle)->comInterface.comHandle24 = NULL;
(*txnHandle)->comInterface.comHandle25 = NULL;
(*txnHandle)->comInterface.comHandle26 = NULL;
(*txnHandle)->comInterface.comHandle27 = NULL;
(*txnHandle)->comInterface.comHandle28 = NULL;
(*txnHandle)->comInterface.comHandle29 = NULL;
(*txnHandle)->comInterface.comHandle30 = NULL;
(*txnHandle)->comInterface.comHandle31 = NULL;
(*txnHandle)->comInterface.comHandle32 = NULL;
(*txnHandle)->comInterface.comHandle33 = NULL;
(*txnHandle)->comInterface.comHandle34 = NULL;
(*txnHandle)->comInterface.comHandle35 = NULL;
(*txnHandle)->comInterface.comHandle36 = NULL;
(*txnHandle)->comInterface.comHandle37 = NULL;
(*txnHandle)->comInterface.comHandle38 = NULL;

```

```

(*txnHandle)->comInterface.comHandle39 = NULL;
(*txnHandle)->comInterface.comHandle40 = NULL;
(*txnHandle)->comInterface.comHandle41 = NULL;
(*txnHandle)->comInterface.comHandle42 = NULL;
(*txnHandle)->comInterface.comHandle43 = NULL;
(*txnHandle)->comInterface.comHandle44 = NULL;
(*txnHandle)->comInterface.comHandle45 = NULL;
(*txnHandle)->comInterface.comHandle46 = NULL;
(*txnHandle)->comInterface.comHandle47 = NULL;
(*txnHandle)->comInterface.comHandle48 = NULL;
(*txnHandle)->comInterface.comHandle49 = NULL;
(*txnHandle)->comInterface.comHandle50 = NULL;
#endif
ERRORMSG("Initializing txnHandle com data buffer to
"<<maxDataSize<<"bytes"<<" numServers "<<numServers<<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: "<<hresults<<endl);
        return(ERR);
    };
    DEBUGMSG("txnHandle com data buffer initialized to " <<
maxDataSize << "bytes" <<endl);
    ERRORMSG("txnHandle com data buffer initialized to " <<
maxDataSize << "bytes " <<"numServers "<<numServers<<endl);

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

    struct _timeb    startTime;
    struct _timeb    endTime;

    DEBUGMSG("Calling CoCreateInstance with
txnHandle:"<<DEBUGADDRESS(*txnHandle)<< endl);
    ERRORMSG("Calling CoCreateInstance with
txnHandle:"<<DEBUGADDRESS(*txnHandle)<< " numServers
"<<numServers<<endl);
    _ftime(&startTime);
    int i;
#ifdef PARTITION
    for (i=0;i<=50;i++) hres[i] = 0;

    if (numServers >= 51) { hres[50] =
CoCreateInstance(CLSID_tpcc_com50,NULL,CLSCTX_SERVER,II
D_Itpcc_com50,(void **)&(*txnHandle)-
>comInterface.comHandle50); }
    if (numServers >= 50) { hres[49] =
CoCreateInstance(CLSID_tpcc_com49,NULL,CLSCTX_SERVER,II
D_Itpcc_com49,(void **)&(*txnHandle)-
>comInterface.comHandle49); }
    if (numServers >= 49) { hres[48] =
CoCreateInstance(CLSID_tpcc_com48,NULL,CLSCTX_SERVER,II
D_Itpcc_com48,(void **)&(*txnHandle)-
>comInterface.comHandle48); }
    if (numServers >= 48) { hres[47] =
CoCreateInstance(CLSID_tpcc_com47,NULL,CLSCTX_SERVER,II
D_Itpcc_com47,(void **)&(*txnHandle)-
>comInterface.comHandle47); }

```

```

if (numServers == 29) { hres[28] =
CoCreateInstance(CLSID_tpcc_com28,NULL,CLSCTX_SERVER,II
D_Itpcc_com28,(void **)&*txnHandle)-
>comInterface.comHandle28); }
if (numServers == 28) { hres[27] =
CoCreateInstance(CLSID_tpcc_com27,NULL,CLSCTX_SERVER,II
D_Itpcc_com27,(void **)&*txnHandle)-
>comInterface.comHandle27); }
if (numServers == 27) { hres[26] =
CoCreateInstance(CLSID_tpcc_com26,NULL,CLSCTX_SERVER,II
D_Itpcc_com26,(void **)&*txnHandle)-
>comInterface.comHandle26); }
if (numServers == 26) { hres[25] =
CoCreateInstance(CLSID_tpcc_com25,NULL,CLSCTX_SERVER,II
D_Itpcc_com25,(void **)&*txnHandle)-
>comInterface.comHandle25); }
if (numServers == 25) { hres[24] =
CoCreateInstance(CLSID_tpcc_com24,NULL,CLSCTX_SERVER,II
D_Itpcc_com24,(void **)&*txnHandle)-
>comInterface.comHandle24); }
if (numServers == 24) { hres[23] =
CoCreateInstance(CLSID_tpcc_com23,NULL,CLSCTX_SERVER,II
D_Itpcc_com23,(void **)&*txnHandle)-
>comInterface.comHandle23); }
if (numServers == 23) { hres[22] =
CoCreateInstance(CLSID_tpcc_com22,NULL,CLSCTX_SERVER,II
D_Itpcc_com22,(void **)&*txnHandle)-
>comInterface.comHandle22); }
if (numServers == 22) { hres[21] =
CoCreateInstance(CLSID_tpcc_com21,NULL,CLSCTX_SERVER,II
D_Itpcc_com21,(void **)&*txnHandle)-
>comInterface.comHandle21); }
if (numServers == 21) { hres[20] =
CoCreateInstance(CLSID_tpcc_com20,NULL,CLSCTX_SERVER,II
D_Itpcc_com20,(void **)&*txnHandle)-
>comInterface.comHandle20); }
if (numServers == 20) { hres[19] =
CoCreateInstance(CLSID_tpcc_com19,NULL,CLSCTX_SERVER,II
D_Itpcc_com19,(void **)&*txnHandle)-
>comInterface.comHandle19); }
if (numServers == 19) { hres[18] =
CoCreateInstance(CLSID_tpcc_com18,NULL,CLSCTX_SERVER,II
D_Itpcc_com18,(void **)&*txnHandle)-
>comInterface.comHandle18); }
if (numServers == 18) { hres[17] =
CoCreateInstance(CLSID_tpcc_com17,NULL,CLSCTX_SERVER,II
D_Itpcc_com17,(void **)&*txnHandle)-
>comInterface.comHandle17); }
if (numServers == 17) { hres[16] =
CoCreateInstance(CLSID_tpcc_com16,NULL,CLSCTX_SERVER,II
D_Itpcc_com16,(void **)&*txnHandle)-
>comInterface.comHandle16); }
if (numServers == 16) { hres[15] =
CoCreateInstance(CLSID_tpcc_com15,NULL,CLSCTX_SERVER,II
D_Itpcc_com15,(void **)&*txnHandle)-
>comInterface.comHandle15); }
if (numServers == 15) { hres[14] =
CoCreateInstance(CLSID_tpcc_com14,NULL,CLSCTX_SERVER,II
D_Itpcc_com14,(void **)&*txnHandle)-
>comInterface.comHandle14); }
if (numServers == 14) { hres[13] =
CoCreateInstance(CLSID_tpcc_com13,NULL,CLSCTX_SERVER,II
D_Itpcc_com13,(void **)&*txnHandle)-
>comInterface.comHandle13); }
if (numServers == 13) { hres[12] =
CoCreateInstance(CLSID_tpcc_com12,NULL,CLSCTX_SERVER,II
D_Itpcc_com12,(void **)&*txnHandle)-
>comInterface.comHandle12); }
if (numServers == 12) { hres[11] =
CoCreateInstance(CLSID_tpcc_com11,NULL,CLSCTX_SERVER,II
D_Itpcc_com11,(void **)&*txnHandle)-
>comInterface.comHandle11); }

```



```

if (numServers >= 11) { hres[10] =
CoCreateInstance(CLSID_tpcc_com10,NULL,CLSCTX_SERVER,II
D_Itpcc_com10,(void **)&(*txnHandle)-
>comInterface.comHandle10); }
if (numServers >= 10) { hres[9] =
CoCreateInstance(CLSID_tpcc_com9,NULL,CLSCTX_SERVER,IID
_Itpcc_com9,(void **)&(*txnHandle)->comInterface.comHandle9); }
if (numServers >= 9) { hres[8] =
CoCreateInstance(CLSID_tpcc_com8,NULL,CLSCTX_SERVER,IID
_Itpcc_com8,(void **)&(*txnHandle)->comInterface.comHandle8); }
if (numServers >= 8) { hres[7] =
CoCreateInstance(CLSID_tpcc_com7,NULL,CLSCTX_SERVER,IID
_Itpcc_com7,(void **)&(*txnHandle)->comInterface.comHandle7); }
if (numServers >= 7) { hres[6] =
CoCreateInstance(CLSID_tpcc_com6,NULL,CLSCTX_SERVER,IID
_Itpcc_com6,(void **)&(*txnHandle)->comInterface.comHandle6); }
if (numServers >= 6) { hres[5] =
CoCreateInstance(CLSID_tpcc_com5,NULL,CLSCTX_SERVER,IID
_Itpcc_com5,(void **)&(*txnHandle)->comInterface.comHandle5); }
if (numServers >= 5) { hres[4] =
CoCreateInstance(CLSID_tpcc_com4,NULL,CLSCTX_SERVER,IID
_Itpcc_com4,(void **)&(*txnHandle)->comInterface.comHandle4); }
if (numServers >= 4) { hres[3] =
CoCreateInstance(CLSID_tpcc_com3,NULL,CLSCTX_SERVER,IID
_Itpcc_com3,(void **)&(*txnHandle)->comInterface.comHandle3); }
if (numServers >= 3) { hres[2] =
CoCreateInstance(CLSID_tpcc_com2,NULL,CLSCTX_SERVER,IID
_Itpcc_com2,(void **)&(*txnHandle)->comInterface.comHandle2); }
if (numServers >= 2) { hres[1] =
CoCreateInstance(CLSID_tpcc_com1,NULL,CLSCTX_SERVER,IID
_Itpcc_com1,(void **)&(*txnHandle)->comInterface.comHandle1); }
#endif
hres[0] =
CoCreateInstance(CLSID_tpcc_com0,NULL,CLSCTX_SERVER,IID
_Itpcc_com0,(void **)&(*txnHandle)->comInterface.comHandle0);
#ifdef PARTITION
for (i=0;i<=50;i++)
#else
for (i=0;i<=0;i++)
{
if (FAILED(hres[i]))
{
ftime(&endTime);
//store error code in txnHandle
ERRORMSG("CoCreateInstance() failed, COM+ Object
"<<i<<," numServers "<<numServers<<," code: "
<<HRESULT_CODE(hres[i])<<"
facility:"<<HRESULT_FACILITY(hres[i])<<
" hres:"<<hex<<hres[i]<<
" time waiting:"<<
(((endTime.time - startTime.time)*1000)+
(endTime.millitm - startTime.millitm))/1000.0)<<endl);

return(ERR);
};
}
#endif
ftime(&endTime);
DEBUGMSG("CoCreateInstance successful.txnHandle com
initialized, time waiting for object to be activated:"<<
(((endTime.time - startTime.time)*1000)+
(endTime.millitm - startTime.millitm))/1000.0)<<endl);
ERRORMSG("CoCreateInstance successful.txnHandle 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.
ERRORMSG("numServers "<<numServers);
#ifdef PARTITION
if (numServers >= 51) { (*txnHandle)->comInterface.comHandle50-
>doSetComplete(); }

```

```

if (numServers >= 50) { (*txnHandle)->comInterface.comHandle49-
>doSetComplete(); }
if (numServers >= 49) { (*txnHandle)->comInterface.comHandle48-
>doSetComplete(); }
if (numServers >= 48) { (*txnHandle)->comInterface.comHandle47-
>doSetComplete(); }
if (numServers >= 47) { (*txnHandle)->comInterface.comHandle46-
>doSetComplete(); }
if (numServers >= 46) { (*txnHandle)->comInterface.comHandle45-
>doSetComplete(); }
if (numServers >= 45) { (*txnHandle)->comInterface.comHandle44-
>doSetComplete(); }
if (numServers >= 44) { (*txnHandle)->comInterface.comHandle43-
>doSetComplete(); }
if (numServers >= 43) { (*txnHandle)->comInterface.comHandle42-
>doSetComplete(); }
if (numServers >= 42) { (*txnHandle)->comInterface.comHandle41-
>doSetComplete(); }
if (numServers >= 41) { (*txnHandle)->comInterface.comHandle40-
>doSetComplete(); }
if (numServers >= 40) { (*txnHandle)->comInterface.comHandle39-
>doSetComplete(); }
if (numServers >= 39) { (*txnHandle)->comInterface.comHandle38-
>doSetComplete(); }
if (numServers >= 38) { (*txnHandle)->comInterface.comHandle37-
>doSetComplete(); }
if (numServers >= 37) { (*txnHandle)->comInterface.comHandle36-
>doSetComplete(); }
if (numServers >= 36) { (*txnHandle)->comInterface.comHandle35-
>doSetComplete(); }
if (numServers >= 35) { (*txnHandle)->comInterface.comHandle34-
>doSetComplete(); }
if (numServers >= 34) { (*txnHandle)->comInterface.comHandle33-
>doSetComplete(); }
if (numServers >= 33) { (*txnHandle)->comInterface.comHandle32-
>doSetComplete(); }
if (numServers >= 32) { (*txnHandle)->comInterface.comHandle31-
>doSetComplete(); }
if (numServers >= 31) { (*txnHandle)->comInterface.comHandle30-
>doSetComplete(); }
if (numServers >= 30) { (*txnHandle)->comInterface.comHandle29-
>doSetComplete(); }
if (numServers >= 29) { (*txnHandle)->comInterface.comHandle28-
>doSetComplete(); }
if (numServers >= 28) { (*txnHandle)->comInterface.comHandle27-
>doSetComplete(); }
if (numServers >= 27) { (*txnHandle)->comInterface.comHandle26-
>doSetComplete(); }
if (numServers >= 26) { (*txnHandle)->comInterface.comHandle25-
>doSetComplete(); }
if (numServers >= 25) { (*txnHandle)->comInterface.comHandle24-
>doSetComplete(); }
if (numServers >= 24) { (*txnHandle)->comInterface.comHandle23-
>doSetComplete(); }
if (numServers >= 23) { (*txnHandle)->comInterface.comHandle22-
>doSetComplete(); }
if (numServers >= 22) { (*txnHandle)->comInterface.comHandle21-
>doSetComplete(); }
if (numServers >= 21) { (*txnHandle)->comInterface.comHandle20-
>doSetComplete(); }
if (numServers >= 20) { (*txnHandle)->comInterface.comHandle19-
>doSetComplete(); }
if (numServers >= 19) { (*txnHandle)->comInterface.comHandle18-
>doSetComplete(); }
if (numServers >= 18) { (*txnHandle)->comInterface.comHandle17-
>doSetComplete(); }
if (numServers >= 17) { (*txnHandle)->comInterface.comHandle16-
>doSetComplete(); }
if (numServers >= 16) { (*txnHandle)->comInterface.comHandle15-
>doSetComplete(); }
if (numServers >= 15) { (*txnHandle)->comInterface.comHandle14-
>doSetComplete(); }

```

```

if (numServers >= 14) { (*txnHandle)->comInterface.comHandle13-
>doSetComplete(); }
if (numServers >= 13) { (*txnHandle)->comInterface.comHandle12-
>doSetComplete(); }
if (numServers >= 12) { (*txnHandle)->comInterface.comHandle11-
>doSetComplete(); }
if (numServers >= 11) { (*txnHandle)->comInterface.comHandle10-
>doSetComplete(); }
if (numServers >= 10) { (*txnHandle)->comInterface.comHandle9-
>doSetComplete(); }
if (numServers >= 9) { (*txnHandle)->comInterface.comHandle8-
>doSetComplete(); }
if (numServers >= 8) { (*txnHandle)->comInterface.comHandle7-
>doSetComplete(); }
if (numServers >= 7) { (*txnHandle)->comInterface.comHandle6-
>doSetComplete(); }
if (numServers >= 6) { (*txnHandle)->comInterface.comHandle5-
>doSetComplete(); }
if (numServers >= 5) { (*txnHandle)->comInterface.comHandle4-
>doSetComplete(); }
if (numServers >= 4) { (*txnHandle)->comInterface.comHandle3-
>doSetComplete(); }
if (numServers >= 3) { (*txnHandle)->comInterface.comHandle2-
>doSetComplete(); }
if (numServers >= 2) { (*txnHandle)->comInterface.comHandle1-
>doSetComplete(); }
#endif
(*txnHandle)->comInterface.comHandle0->doSetComplete();

    ERRORMSG("doSetComplete Complete."<<endl);
    //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 registry
**          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
    {
        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;
    }
    ERRORMSG("Found all addresses in dll"<<endl);
    return OK;
}

```

```

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

int initDlvy()
{
    ERRORMSG(">>initDlvy"<<endl);
    // Initialize critical 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 critical 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 available 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      : readRegistryValues
** Description :
**      initialize isapi global variables from registry
** Parameters:
**
** Returns   :
**      int - return code
** Comments  :
**
*****
*/
int readRegistryValues()
{
    HKEY registryKey;
    char value[MAX_STRING_LEN];
    DWORD regType;
    DWORD regValue;
    DWORD regValueSize = MAX_STRING_LEN;

    // ERRORMSG(">>readRegistryValues"<<endl);
    //open up registry key
    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTRY_SUB_
    KEY,0,KEY_READ,&registryKey) != ERROR_SUCCESS)
        return ERR_UNABLE_TO_OPEN_REG;

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

    //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);

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

    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registryKey,"numServers",0,&regType,(BYT
    E *)&regValue,&regValueSize) == ERROR_SUCCESS)

```

```

    numServers = regValue;
else
    return ERR_NUMSERVERS_NOT_IN_REG;

    ERRORMSG("numServers="<<numServers<<endl);
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registryKey,"numWarehouse",0,&regType,(
    BYTE *)&regValue,&regValueSize) == ERROR_SUCCESS)
        numWarehouse = regValue;
    else
        return ERR_NUMWAREHOUSES_NOT_IN_REG;

    ERRORMSG("numWarehouse="<<numWarehouse<<endl);
    //get num dlvy threads
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registryKey,DELIVERY_THREADS,0,&regT
ype,(BYTE *)&regValue,&regValueSize) == ERROR_SUCCESS)
        dlvyThreads = regValue;
    else
        dlvyThreads = DEFAULT_DLvy_THREADS;

    //get dlvy queue len
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registryKey,DELIVERY_QUEUE_LEN,0,&r
egType,(BYTE *)&regValue,&regValueSize) ==
    ERROR_SUCCESS)
        dlvyQueueLen = regValue;
    else
        dlvyQueueLen = DEFAULT_DLvy_QUEUE_LEN;

    //get the htmlTrace flag
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registryKey,HTML_TRACE,0,&regType,(BY
TE *)&regValue,&regValueSize) == ERROR_SUCCESS)
        trace = regValue;
    else
        trace = 0;

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

    //get the num of users
    regValueSize = sizeof(regValue);
    if(RegQueryValueEx(registryKey,NUM_USERS,0,&regType,(BY
TE *)&regValue,&regValueSize) == ERROR_SUCCESS)
        numUsers = regValue;
    else
        numUsers = DEFAULT_NUM_USERS;

    //get dlvy log file path
    regValueSize = sizeof(value);
    if
    (RegQueryValueEx(registryKey,DELIVERY_LOG_PATH,0,&regTy
pe,(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(registryKey,HTML_TRACE_LOG_FILE,0,&reg
Type,(BYTE *) &value,&regValueSize) == ERROR_SUCCESS )
        strcpy(htmlTraceLogFile,value);
    else

    strcpy(htmlTraceLogFile,DEFAULT_HTML_TRACE_LOG_FILE);

```

```

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

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

RegCloseKey(registryKey);
double ware=numWarehouse,serv=numServers;
ratio = (int)ceil((double)(ware/serv));
ERRORMSG("ratio="<<ratio<<endl);
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(dlvBufferThreadIndex, 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"
//
"123456789012345678901234567890123456789012345678901234567890123
4567890123456789012345678901234567890\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"
//
"123456789012345678901234567890123456789012345678901234567890123
4567890123456789012345678901234567890\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,maxDataSize);
    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,commandBlock
,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,commandBlock
,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)
#else 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)
#endif
{
doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_ID,com
mandBlock,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)
#else ORACLE
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,comman
dBlock,txnHandle);
return OK;
}
else
itemComplete++;
}
//missing previous value of item supp warehouse, flag error
else
{
doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_ID,com
mandBlock,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,comman
dBlock,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)
#endif
        {
            doNewOrderErrorPage(html,ERR_INVALID_ITEM_OTY,comman
dBlock,txnHandle);
            return OK;
        }
        else
            itemComplete++;
    }
    //missing previous value of item number
    else if (itemComplete ==1)
    {
        doNewOrderErrorPage(html,ERR_INVALID_ITEM_NUM,comman
dBlock,txnHandle);
        return OK;
    }
    //missing 1st value of supp warehouse
    else
    {
        doNewOrderErrorPage(html,ERR_INVALID_SUPPLY_W_ID,com
mandBlock,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,comman
dBlock,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;
#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"
//
"12345678901234567890123456789012345678901234567890123
4567890123456789012345678901234567890\r\n
\"");
//assume failure
nord->out_nord.s_transtatus = -1;

DEBUGMSG("nord executing COM interface function" << endl);
HRESULT hres;
try
{
#ifdef PARTITION
    switch (txnHandle->part_server)
    {
case 0:hres = txnHandle->comInterface.comHandle0-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 1:hres = txnHandle->comInterface.comHandle1-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 2:hres = txnHandle->comInterface.comHandle2-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 3:hres = txnHandle->comInterface.comHandle3-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 4:hres = txnHandle->comInterface.comHandle4-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 5:hres = txnHandle->comInterface.comHandle5-
>doNewOrder(&txnHandle-

```



```

>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 42:hres = txnHandle->comInterface.comHandle42-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 43:hres = txnHandle->comInterface.comHandle43-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 44:hres = txnHandle->comInterface.comHandle44-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 45:hres = txnHandle->comInterface.comHandle45-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 46:hres = txnHandle->comInterface.comHandle46-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 47:hres = txnHandle->comInterface.comHandle47-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 48:hres = txnHandle->comInterface.comHandle48-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 49:hres = txnHandle->comInterface.comHandle49-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 50:hres = txnHandle->comInterface.comHandle50-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
}
#else
hres = txnHandle->comInterface.comHandle0-
>doNewOrder(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);
#endif
}
catch(...)
{
#ifdef PARTITION
html+=sprintf(html,"ERROR : nord com call caused exeception,
warehouse: %d, COM server %d, ratio
%d</PRE></BODY></HTML>",nord->in_nord.s_W_ID,txnHandle-
>part_server,ratio);
#else
html+=sprintf(html,"ERROR : nord com call caused exeception,
warehouse: %d</PRE></BODY></HTML>",nord->in_nord.s_W_ID);
#endif
return OK;
}

if(FAILED(hres))
{
    ERRORMSG("ERROR : nord com call failed, rc: " <<
DEBUGADDRESS(hex) << hres);
    DEBUGMSG("ERROR : nord com call failed, rc: " << hex <<
hres);
}
else
{
    //com call successful, return object back to pool.
#ifdef PARTITION
switch (txnHandle->part_server) {

```

```

case 0: hres = txnHandle->comInterface.comHandle0-
>doSetComplete();break;
case 1: hres = txnHandle->comInterface.comHandle1-
>doSetComplete();break;
case 2: hres = txnHandle->comInterface.comHandle2-
>doSetComplete();break;
case 3: hres = txnHandle->comInterface.comHandle3-
>doSetComplete();break;
case 4: hres = txnHandle->comInterface.comHandle4-
>doSetComplete();break;
case 5: hres = txnHandle->comInterface.comHandle5-
>doSetComplete();break;
case 6: hres = txnHandle->comInterface.comHandle6-
>doSetComplete();break;
case 7: hres = txnHandle->comInterface.comHandle7-
>doSetComplete();break;
case 8: hres = txnHandle->comInterface.comHandle8-
>doSetComplete();break;
case 9: hres = txnHandle->comInterface.comHandle9-
>doSetComplete();break;
case 10: hres = txnHandle->comInterface.comHandle10-
>doSetComplete();break;
case 11: hres = txnHandle->comInterface.comHandle11-
>doSetComplete();break;
case 12: hres = txnHandle->comInterface.comHandle12-
>doSetComplete();break;
case 13: hres = txnHandle->comInterface.comHandle13-
>doSetComplete();break;
case 14: hres = txnHandle->comInterface.comHandle14-
>doSetComplete();break;
case 15: hres = txnHandle->comInterface.comHandle15-
>doSetComplete();break;
case 16: hres = txnHandle->comInterface.comHandle16-
>doSetComplete();break;
case 17: hres = txnHandle->comInterface.comHandle17-
>doSetComplete();break;
case 18: hres = txnHandle->comInterface.comHandle18-
>doSetComplete();break;
case 19: hres = txnHandle->comInterface.comHandle19-
>doSetComplete();break;
case 20: hres = txnHandle->comInterface.comHandle20-
>doSetComplete();break;
case 21: hres = txnHandle->comInterface.comHandle21-
>doSetComplete();break;
case 22: hres = txnHandle->comInterface.comHandle22-
>doSetComplete();break;
case 23: hres = txnHandle->comInterface.comHandle23-
>doSetComplete();break;
case 24: hres = txnHandle->comInterface.comHandle24-
>doSetComplete();break;
case 25: hres = txnHandle->comInterface.comHandle25-
>doSetComplete();break;
case 26: hres = txnHandle->comInterface.comHandle26-
>doSetComplete();break;
case 27: hres = txnHandle->comInterface.comHandle27-
>doSetComplete();break;
case 28: hres = txnHandle->comInterface.comHandle28-
>doSetComplete();break;
case 29: hres = txnHandle->comInterface.comHandle29-
>doSetComplete();break;
case 30: hres = txnHandle->comInterface.comHandle30-
>doSetComplete();break;
case 31: hres = txnHandle->comInterface.comHandle31-
>doSetComplete();break;
case 32: hres = txnHandle->comInterface.comHandle32-
>doSetComplete();break;
case 33: hres = txnHandle->comInterface.comHandle33-
>doSetComplete();break;
case 34: hres = txnHandle->comInterface.comHandle34-
>doSetComplete();break;
case 35: hres = txnHandle->comInterface.comHandle35-
>doSetComplete();break;

```

```

case 36: hres = txnHandle->comInterface.comHandle36-
>doSetComplete();break;
case 37: hres = txnHandle->comInterface.comHandle37-
>doSetComplete();break;
case 38: hres = txnHandle->comInterface.comHandle38-
>doSetComplete();break;
case 39: hres = txnHandle->comInterface.comHandle39-
>doSetComplete();break;
case 40: hres = txnHandle->comInterface.comHandle40-
>doSetComplete();break;
case 41: hres = txnHandle->comInterface.comHandle41-
>doSetComplete();break;
case 42: hres = txnHandle->comInterface.comHandle42-
>doSetComplete();break;
case 43: hres = txnHandle->comInterface.comHandle43-
>doSetComplete();break;
case 44: hres = txnHandle->comInterface.comHandle44-
>doSetComplete();break;
case 45: hres = txnHandle->comInterface.comHandle45-
>doSetComplete();break;
case 46: hres = txnHandle->comInterface.comHandle46-
>doSetComplete();break;
case 47: hres = txnHandle->comInterface.comHandle47-
>doSetComplete();break;
case 48: hres = txnHandle->comInterface.comHandle48-
>doSetComplete();break;
case 49: hres = txnHandle->comInterface.comHandle49-
>doSetComplete();break;
case 50: hres = txnHandle->comInterface.comHandle50-
>doSetComplete();break;
}
#else
hres = txnHandle->comInterface.comHandle0->doSetComplete();
#endif
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: ");
if(rc == OK)
{
#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);
#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);
#elif DB2
        html+=sprintf(html,"%2.2lf",nord-
>out_nord.s_C_DISCOUNT/100.0);
#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);
#elif DB2
        html+=sprintf(html,"%5.2lf",nord->out_nord.s_W_TAX/100.0);
#endif
    }

    appendText(&html," D_Tax: ");
    if(rc == OK)
    {
#ifdef ORACLE
        html+=sprintf(html,"%5.2lf",nord->out_nord.s_D_TAX);
#elif DB2
        html+=sprintf(html,"%5.2lf",nord->out_nord.s_D_TAX/100.0);
#endif
    }
    appendText(&html," <BR> <BR>"
// " 1 2 3 4 5 6 7 8
9\r\n"
//
"123456789012345678901234567890123456789012345678901234567890123
4567890123456789012345678901234567890\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_LEN+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);
#ifdef 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);
#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>");

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

    html+=displayStatus(html, rc);
    if(rc == OK)
#ifdef ORACLE
        html+=sprintf(html, "    Total: $%-7.2lf", nord-
>out_nord.s_total_amount/100.0);
#elseif DB2
        html+=sprintf(html, "    Total: $%-7.2lf", nord-
>out_nord.s_total_amount);
#endif
    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>\n\n"
        "<BODY><FORM ACTION=\"\"
        APP_NAME
        \"\" METHOD=\"GET\">\n\n"
        "<CENTER><H3>Please Fill In New Order
Form.</H3></CENTER>\n\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\n\n"

"123456789012345678901234567890123456789012345678901234567890123
45678901234567890123456789012345678901234567890\n\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\n\n"

//"12345678901234567890123456789012345678901234567890123456789012
34567890123456789012345678901234567890\n\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

```



```

    if ( (pymt->in_paym.s_C_D_ID = atoi(commandBlock-
>get_C_D_ID())) == 0)
    {

        doPaymentErrorPage(html,ERR_INVALID_C_D_ID,commandBloc
k,txnHandle);
        return OK;
    }

    if(!copyInMoney32(commandBlock->get_AMT_PAID(),&pymt-
>in_paym.s_H_AMOUNT))
    {

        doPaymentErrorPage(html,ERR_INVALID_PAYMENT_AMOUNT,
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></C
ENTER>");

    DEBUGMSG("Calling com entry api payment, w_id:<<pymt-
>in_paym.s_W_ID<<" d_id:<<pymt->in_paym.s_D_ID<<endl);
    //assume failure
    pymt->out_paym.s_transtatus = -1;

    HRESULT hres;
    try
    {
#ifdef PARTITION
        switch (txnHandle->part_server)
        {
case 0:hres = txnHandle->comInterface.comHandle0-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 1:hres = txnHandle->comInterface.comHandle1-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 2:hres = txnHandle->comInterface.comHandle2-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 3:hres = txnHandle->comInterface.comHandle3-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 4:hres = txnHandle->comInterface.comHandle4-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 5:hres = txnHandle->comInterface.comHandle5-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 6:hres = txnHandle->comInterface.comHandle6-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 7:hres = txnHandle->comInterface.comHandle7-
>doPayment(&txnHandle-

```

```

>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 8:hres = txnHandle->comInterface.comHandle8-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 9:hres = txnHandle->comInterface.comHandle9-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 10:hres = txnHandle->comInterface.comHandle10-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 11:hres = txnHandle->comInterface.comHandle11-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 12:hres = txnHandle->comInterface.comHandle12-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 13:hres = txnHandle->comInterface.comHandle13-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 14:hres = txnHandle->comInterface.comHandle14-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 15:hres = txnHandle->comInterface.comHandle15-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 16:hres = txnHandle->comInterface.comHandle16-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 17:hres = txnHandle->comInterface.comHandle17-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 18:hres = txnHandle->comInterface.comHandle18-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 19:hres = txnHandle->comInterface.comHandle19-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 20:hres = txnHandle->comInterface.comHandle20-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 21:hres = txnHandle->comInterface.comHandle21-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 22:hres = txnHandle->comInterface.comHandle22-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 23:hres = txnHandle->comInterface.comHandle23-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 24:hres = txnHandle->comInterface.comHandle24-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR*)&txnHandle-
>comInterface.txnBuffer);break;
case 25:hres = txnHandle->comInterface.comHandle25-
>doPayment(&txnHandle-

```

```

>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 26:hres = txnHandle->comInterface.comHandle26-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 27:hres = txnHandle->comInterface.comHandle27-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 28:hres = txnHandle->comInterface.comHandle28-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 29:hres = txnHandle->comInterface.comHandle29-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 30:hres = txnHandle->comInterface.comHandle30-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 31:hres = txnHandle->comInterface.comHandle31-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 32:hres = txnHandle->comInterface.comHandle32-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 33:hres = txnHandle->comInterface.comHandle33-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 34:hres = txnHandle->comInterface.comHandle34-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 35:hres = txnHandle->comInterface.comHandle35-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 36:hres = txnHandle->comInterface.comHandle36-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 37:hres = txnHandle->comInterface.comHandle37-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 38:hres = txnHandle->comInterface.comHandle38-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 39:hres = txnHandle->comInterface.comHandle39-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 40:hres = txnHandle->comInterface.comHandle40-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 41:hres = txnHandle->comInterface.comHandle41-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 42:hres = txnHandle->comInterface.comHandle42-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 43:hres = txnHandle->comInterface.comHandle43-
>doPayment(&txnHandle-

```

```

>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 44:hres = txnHandle->comInterface.comHandle44-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 45:hres = txnHandle->comInterface.comHandle45-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 46:hres = txnHandle->comInterface.comHandle46-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 47:hres = txnHandle->comInterface.comHandle47-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 48:hres = txnHandle->comInterface.comHandle48-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 49:hres = txnHandle->comInterface.comHandle49-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break;
case 50:hres = txnHandle->comInterface.comHandle50-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);break; }
#else
hres = txnHandle->comInterface.comHandle0-
>doPayment(&txnHandle-
>comInterface.size,(UCHAR**) &txnHandle-
>comInterface.txnBuffer);
#endif
}
catch(...)
{
#ifdef PARTITION
    html+=sprintf(html,"ERROR : Com pymt call caused exeception,
warehouse: %d, COM server %d, ratio
%d</PRE></BODY></HTML>",pymt->in_paym.s_W_ID,txnHandle-
>part_server,ratio);
#else
    html+=sprintf(html,"ERROR : Com pymt call caused exeception,
warehouse: %d</PRE></BODY></HTML>",pymt-
>in_paym.s_W_ID);
#endif
    return OK;
}

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

#ifdef PARTITION
    switch (txnHandle->part_server) {
case 0: hres = txnHandle->comInterface.comHandle0-
>doSetComplete();break;
case 1: hres = txnHandle->comInterface.comHandle1-
>doSetComplete();break;
case 2: hres = txnHandle->comInterface.comHandle2-
>doSetComplete();break;
case 3: hres = txnHandle->comInterface.comHandle3-
>doSetComplete();break;

```

```

case 4: hres = txnHandle->comInterface.comHandle4-
>doSetComplete();break;
case 5: hres = txnHandle->comInterface.comHandle5-
>doSetComplete();break;
case 6: hres = txnHandle->comInterface.comHandle6-
>doSetComplete();break;
case 7: hres = txnHandle->comInterface.comHandle7-
>doSetComplete();break;
case 8: hres = txnHandle->comInterface.comHandle8-
>doSetComplete();break;
case 9: hres = txnHandle->comInterface.comHandle9-
>doSetComplete();break;
case 10: hres = txnHandle->comInterface.comHandle10-
>doSetComplete();break;
case 11: hres = txnHandle->comInterface.comHandle11-
>doSetComplete();break;
case 12: hres = txnHandle->comInterface.comHandle12-
>doSetComplete();break;
case 13: hres = txnHandle->comInterface.comHandle13-
>doSetComplete();break;
case 14: hres = txnHandle->comInterface.comHandle14-
>doSetComplete();break;
case 15: hres = txnHandle->comInterface.comHandle15-
>doSetComplete();break;
case 16: hres = txnHandle->comInterface.comHandle16-
>doSetComplete();break;
case 17: hres = txnHandle->comInterface.comHandle17-
>doSetComplete();break;
case 18: hres = txnHandle->comInterface.comHandle18-
>doSetComplete();break;
case 19: hres = txnHandle->comInterface.comHandle19-
>doSetComplete();break;
case 20: hres = txnHandle->comInterface.comHandle20-
>doSetComplete();break;
case 21: hres = txnHandle->comInterface.comHandle21-
>doSetComplete();break;
case 22: hres = txnHandle->comInterface.comHandle22-
>doSetComplete();break;
case 23: hres = txnHandle->comInterface.comHandle23-
>doSetComplete();break;
case 24: hres = txnHandle->comInterface.comHandle24-
>doSetComplete();break;
case 25: hres = txnHandle->comInterface.comHandle25-
>doSetComplete();break;
case 26: hres = txnHandle->comInterface.comHandle26-
>doSetComplete();break;
case 27: hres = txnHandle->comInterface.comHandle27-
>doSetComplete();break;
case 28: hres = txnHandle->comInterface.comHandle28-
>doSetComplete();break;
case 29: hres = txnHandle->comInterface.comHandle29-
>doSetComplete();break;
case 30: hres = txnHandle->comInterface.comHandle30-
>doSetComplete();break;
case 31: hres = txnHandle->comInterface.comHandle31-
>doSetComplete();break;
case 32: hres = txnHandle->comInterface.comHandle32-
>doSetComplete();break;
case 33: hres = txnHandle->comInterface.comHandle33-
>doSetComplete();break;
case 34: hres = txnHandle->comInterface.comHandle34-
>doSetComplete();break;
case 35: hres = txnHandle->comInterface.comHandle35-
>doSetComplete();break;
case 36: hres = txnHandle->comInterface.comHandle36-
>doSetComplete();break;
case 37: hres = txnHandle->comInterface.comHandle37-
>doSetComplete();break;
case 38: hres = txnHandle->comInterface.comHandle38-
>doSetComplete();break;
case 39: hres = txnHandle->comInterface.comHandle39-
>doSetComplete();break;

```

```

case 40: hres = txnHandle->comInterface.comHandle40-
>doSetComplete();break;
case 41: hres = txnHandle->comInterface.comHandle41-
>doSetComplete();break;
case 42: hres = txnHandle->comInterface.comHandle42-
>doSetComplete();break;
case 43: hres = txnHandle->comInterface.comHandle43-
>doSetComplete();break;
case 44: hres = txnHandle->comInterface.comHandle44-
>doSetComplete();break;
case 45: hres = txnHandle->comInterface.comHandle45-
>doSetComplete();break;
case 46: hres = txnHandle->comInterface.comHandle46-
>doSetComplete();break;
case 47: hres = txnHandle->comInterface.comHandle47-
>doSetComplete();break;
case 48: hres = txnHandle->comInterface.comHandle48-
>doSetComplete();break;
case 49: hres = txnHandle->comInterface.comHandle49-
>doSetComplete();break;
case 50: hres = txnHandle->comInterface.comHandle50-
>doSetComplete();break;
}
#else
hres = txnHandle->comInterface.comHandle0->doSetComplete();
#endif
pymt = (pymt_wrapper *)txnHandle->comInterface.txnBuffer;

//get return code
int rc = pymt->out_paym.s_transtatus;
if( rc != OK)
{
    html+=displayStatus(html,rc);
    appendText(&html,itoa(rc,buffer,10));
    appendText(&html,"</PRE></BODY></HTML>");
    return OK;
}
// appendText(&html, "<BR><PRE>\n\n");
// appendText(&html, "      1      2      3      4      5      6
7      8<BR>");
//
appendText(&html,"12345678901234567890123456789012345678901234567
8901234567890123456789012345678901234567890<BR>");

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

#ifdef ORACLE
    appendText(&html,pymt->out_paym.s_H_DATE_time);
#endif
#ifdef DB2
    copyOutDateTime(buffer,pymt->out_paym.s_H_DATE_time);
    appendText(&html,buffer);
#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);
#endif
#ifdef DB2
    copyOutDateTime(buffer,pymt->out_paym.s_C_SINCE_time);
    appendText(&html,buffer);
#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: ");
#ifdef ORACLE
    html+=sprintf(html,"%2.2lf",pymt->out_paym.s_C_DISCOUNT);
#elif DB2
    html+=sprintf(html,"%2.2lf",pymt-
>out_paym.s_C_DISCOUNT/100.0);
#endif
    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: $");
#ifdef ORACLE
    html+=sprintf(html,"%-9.2lf",pymt->out_paym.s_C_BALANCE);
#elif DB2
    html+=sprintf(html,"%-9.2lf",pymt-
>out_paym.s_C_BALANCE/100.0);
#endif
    appendText(&html,"<BR>Credit Limit: $");
#ifdef ORACLE
    html+=sprintf(html,"%-9.2lf",pymt->out_paym.s_C_CREDIT_LIM);
#elif DB2
    html+=sprintf(html,"%-9.2lf",pymt-
>out_paym.s_C_CREDIT_LIM/100.0);
#endif
    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,ittoa(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,ittoa(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   Qty      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,maxDataSize);

    //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,commandBlock,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_ONLY,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 = -1;

HRESULT hres;
try
{
#ifdef PARTITION
    switch (txnHandle->part_server)
    {
case 0:hres = txnHandle->comInterface.comHandle0->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 1:hres = txnHandle->comInterface.comHandle1->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 2:hres = txnHandle->comInterface.comHandle2->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 3:hres = txnHandle->comInterface.comHandle3->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 4:hres = txnHandle->comInterface.comHandle4->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;

```

```

case 5:hres = txnHandle->comInterface.comHandle5->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 6:hres = txnHandle->comInterface.comHandle6->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 7:hres = txnHandle->comInterface.comHandle7->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 8:hres = txnHandle->comInterface.comHandle8->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 9:hres = txnHandle->comInterface.comHandle9->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 10:hres = txnHandle->comInterface.comHandle10->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 11:hres = txnHandle->comInterface.comHandle11->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 12:hres = txnHandle->comInterface.comHandle12->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 13:hres = txnHandle->comInterface.comHandle13->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 14:hres = txnHandle->comInterface.comHandle14->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 15:hres = txnHandle->comInterface.comHandle15->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 16:hres = txnHandle->comInterface.comHandle16->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 17:hres = txnHandle->comInterface.comHandle17->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 18:hres = txnHandle->comInterface.comHandle18->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 19:hres = txnHandle->comInterface.comHandle19->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 20:hres = txnHandle->comInterface.comHandle20->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 21:hres = txnHandle->comInterface.comHandle21->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;
case 22:hres = txnHandle->comInterface.comHandle22->doOrderStatus(&txnHandle->comInterface.size,(UCHAR**)&txnHandle->comInterface.txnBuffer);break;

```

```

case 23:hres = txnHandle->comInterface.comHandle23-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 24:hres = txnHandle->comInterface.comHandle24-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 25:hres = txnHandle->comInterface.comHandle25-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 26:hres = txnHandle->comInterface.comHandle26-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 27:hres = txnHandle->comInterface.comHandle27-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 28:hres = txnHandle->comInterface.comHandle28-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 29:hres = txnHandle->comInterface.comHandle29-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 30:hres = txnHandle->comInterface.comHandle30-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 31:hres = txnHandle->comInterface.comHandle31-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 32:hres = txnHandle->comInterface.comHandle32-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 33:hres = txnHandle->comInterface.comHandle33-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 34:hres = txnHandle->comInterface.comHandle34-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 35:hres = txnHandle->comInterface.comHandle35-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 36:hres = txnHandle->comInterface.comHandle36-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 37:hres = txnHandle->comInterface.comHandle37-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 38:hres = txnHandle->comInterface.comHandle38-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 39:hres = txnHandle->comInterface.comHandle39-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 40:hres = txnHandle->comInterface.comHandle40-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;

```

```

case 41:hres = txnHandle->comInterface.comHandle41-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 42:hres = txnHandle->comInterface.comHandle42-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 43:hres = txnHandle->comInterface.comHandle43-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 44:hres = txnHandle->comInterface.comHandle44-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 45:hres = txnHandle->comInterface.comHandle45-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 46:hres = txnHandle->comInterface.comHandle46-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 47:hres = txnHandle->comInterface.comHandle47-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 48:hres = txnHandle->comInterface.comHandle48-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 49:hres = txnHandle->comInterface.comHandle49-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 50:hres = txnHandle->comInterface.comHandle50-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
    }
#else
hres = txnHandle->comInterface.comHandle0-
>doOrderStatus(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);
#endif
    }
    catch(...)
    {
#ifdef PARTITION
        html+=sprintf(html,"ERROR : ords com call caused exeception,
warehouse: %d, COM server %d, ratio
%d.</PRE></BODY></HTML>",ords->in_ords.s_W_ID,txnHandle-
>part_server,ratio);
#else
        html+=sprintf(html,"ERROR : ords com call caused exeception,
warehouse: %d</PRE></BODY></HTML>",ords->in_ords.s_W_ID);
#endif
        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));
        return OK;
    }

#ifdef PARTITION

```

```

switch (txnHandle->part_server) {
case 0: hres = txnHandle->comInterface.comHandle0->doSetComplete();break;
case 1: hres = txnHandle->comInterface.comHandle1->doSetComplete();break;
case 2: hres = txnHandle->comInterface.comHandle2->doSetComplete();break;
case 3: hres = txnHandle->comInterface.comHandle3->doSetComplete();break;
case 4: hres = txnHandle->comInterface.comHandle4->doSetComplete();break;
case 5: hres = txnHandle->comInterface.comHandle5->doSetComplete();break;
case 6: hres = txnHandle->comInterface.comHandle6->doSetComplete();break;
case 7: hres = txnHandle->comInterface.comHandle7->doSetComplete();break;
case 8: hres = txnHandle->comInterface.comHandle8->doSetComplete();break;
case 9: hres = txnHandle->comInterface.comHandle9->doSetComplete();break;
case 10: hres = txnHandle->comInterface.comHandle10->doSetComplete();break;
case 11: hres = txnHandle->comInterface.comHandle11->doSetComplete();break;
case 12: hres = txnHandle->comInterface.comHandle12->doSetComplete();break;
case 13: hres = txnHandle->comInterface.comHandle13->doSetComplete();break;
case 14: hres = txnHandle->comInterface.comHandle14->doSetComplete();break;
case 15: hres = txnHandle->comInterface.comHandle15->doSetComplete();break;
case 16: hres = txnHandle->comInterface.comHandle16->doSetComplete();break;
case 17: hres = txnHandle->comInterface.comHandle17->doSetComplete();break;
case 18: hres = txnHandle->comInterface.comHandle18->doSetComplete();break;
case 19: hres = txnHandle->comInterface.comHandle19->doSetComplete();break;
case 20: hres = txnHandle->comInterface.comHandle20->doSetComplete();break;
case 21: hres = txnHandle->comInterface.comHandle21->doSetComplete();break;
case 22: hres = txnHandle->comInterface.comHandle22->doSetComplete();break;
case 23: hres = txnHandle->comInterface.comHandle23->doSetComplete();break;
case 24: hres = txnHandle->comInterface.comHandle24->doSetComplete();break;
case 25: hres = txnHandle->comInterface.comHandle25->doSetComplete();break;
case 26: hres = txnHandle->comInterface.comHandle26->doSetComplete();break;
case 27: hres = txnHandle->comInterface.comHandle27->doSetComplete();break;
case 28: hres = txnHandle->comInterface.comHandle28->doSetComplete();break;
case 29: hres = txnHandle->comInterface.comHandle29->doSetComplete();break;
case 30: hres = txnHandle->comInterface.comHandle30->doSetComplete();break;
case 31: hres = txnHandle->comInterface.comHandle31->doSetComplete();break;
case 32: hres = txnHandle->comInterface.comHandle32->doSetComplete();break;
case 33: hres = txnHandle->comInterface.comHandle33->doSetComplete();break;
case 34: hres = txnHandle->comInterface.comHandle34->doSetComplete();break;

```

```

case 35: hres = txnHandle->comInterface.comHandle35->doSetComplete();break;
case 36: hres = txnHandle->comInterface.comHandle36->doSetComplete();break;
case 37: hres = txnHandle->comInterface.comHandle37->doSetComplete();break;
case 38: hres = txnHandle->comInterface.comHandle38->doSetComplete();break;
case 39: hres = txnHandle->comInterface.comHandle39->doSetComplete();break;
case 40: hres = txnHandle->comInterface.comHandle40->doSetComplete();break;
case 41: hres = txnHandle->comInterface.comHandle41->doSetComplete();break;
case 42: hres = txnHandle->comInterface.comHandle42->doSetComplete();break;
case 43: hres = txnHandle->comInterface.comHandle43->doSetComplete();break;
case 44: hres = txnHandle->comInterface.comHandle44->doSetComplete();break;
case 45: hres = txnHandle->comInterface.comHandle45->doSetComplete();break;
case 46: hres = txnHandle->comInterface.comHandle46->doSetComplete();break;
case 47: hres = txnHandle->comInterface.comHandle47->doSetComplete();break;
case 48: hres = txnHandle->comInterface.comHandle48->doSetComplete();break;
case 49: hres = txnHandle->comInterface.comHandle49->doSetComplete();break;
case 50: hres = txnHandle->comInterface.comHandle50->doSetComplete();break;
}
#else
hres = txnHandle->comInterface.comHandle0->doSetComplete();
#endif
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>\n\n");
// appendText(&html,"      1      2      3      4      5      6
7      8<BR>");
//
appendText(&html,"12345678901234567890123456789012345678901234567
8901234567890123456789012345678901234567890<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: $");
    html+=sprintf(html,"%0.2lf",ords->out_ords.s_C_BALANCE/100.0);

    //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),12,1);
    appendText(&html,"Entry-Date: ");
#ifdef ORACLE
    appendText(&html,ords->out_ords.s_O_ENTRY_D_time);
#endif
#ifdef DB2
    copyOutDateTime(buffer,ords->out_ords.s_O_ENTRY_D_time);
    appendText(&html,buffer,22,1);
#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 qty
        appendText(&html,itoa(ords-
>out_ords.item[itemCount].s_OL_QUANTITY,buffer,10),6,1);

        //get item dollar amount
        html+=sprintf(html,"%0-14.2lf",ords-
>out_ords.item[itemCount].s_OL_AMOUNT/100.0);
#else 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 qty
        appendText(&html,itoa(ords-
>out_ords.s_OL_QUANTITY[itemCount],buffer,10),6,1);

        //get item dollar amount
        html+=sprintf(html,"%0-14.2lf",ords-
>out_ords.s_OL_AMOUNT[itemCount]/100.0);
#endif

        //get delivery date

```

```

#ifdef ORACLE
    appendText(&html,ords-
>out_ords.s_OL_DELIVERY_D_time[itemCount]);
#endif
#ifdef DB2
    copyOutDate(buffer,ords-
>out_ords.item[itemCount].s_OL_DELIVERY_D_time);
    appendText(&html,buffer);
#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  Qty      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\\""
        "<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_dlv.s_W_ID = txnHandle->w_id;

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

            doDeliveryErrorPage(html,ERR_INVALID_CARRIER,commandBl
ock,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\\""
        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_dlv.s_W_ID,dlvy.in_dlv.s_O_CARRIER_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_dlv.s_W_ID,buffer,10));
        appendText(&html,"<BR> <BR>Carrier Number: ");

        //get carrier_id from wrapper
        appendText(&html,itoa(dlvy.in_dlv.s_O_CARRIER_ID,buffer,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),6+1,1);
    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      : 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,maxDataSize);

    //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,commandB
lock,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>\n\n"
        "<BODY><FORM ACTION=\""
        APP_NAME
        "\" METHOD=\""GET\"">\n\n");

    html+=appendButtons(html);

    html+=appendHiddenFields(html,txnHandle);

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

    stok->out_stok.s_transtatus = -1;

    DEBUGMSG("Calling com entry api for stock call, w_id:"<<stok-
>in_stok.s_W_ID<<" d_id:"<<stok->in_stok.s_D_ID<<
    " threshold:"<<stok->in_stok.s_threshold<<endl);

    HRESULT hres;
    try
    {
    #ifdef PARTITION
        switch (txnHandle->part_server)
        {
        case 0:hres = txnHandle->comInterface.comHandle0-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 1:hres = txnHandle->comInterface.comHandle1-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 2:hres = txnHandle->comInterface.comHandle2-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 3:hres = txnHandle->comInterface.comHandle3-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 4:hres = txnHandle->comInterface.comHandle4-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 5:hres = txnHandle->comInterface.comHandle5-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 6:hres = txnHandle->comInterface.comHandle6-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 7:hres = txnHandle->comInterface.comHandle7-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 8:hres = txnHandle->comInterface.comHandle8-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;

```

```

        case 9:hres = txnHandle->comInterface.comHandle9-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 10:hres = txnHandle->comInterface.comHandle10-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 11:hres = txnHandle->comInterface.comHandle11-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 12:hres = txnHandle->comInterface.comHandle12-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 13:hres = txnHandle->comInterface.comHandle13-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 14:hres = txnHandle->comInterface.comHandle14-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 15:hres = txnHandle->comInterface.comHandle15-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 16:hres = txnHandle->comInterface.comHandle16-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 17:hres = txnHandle->comInterface.comHandle17-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 18:hres = txnHandle->comInterface.comHandle18-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 19:hres = txnHandle->comInterface.comHandle19-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 20:hres = txnHandle->comInterface.comHandle20-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 21:hres = txnHandle->comInterface.comHandle21-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 22:hres = txnHandle->comInterface.comHandle22-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 23:hres = txnHandle->comInterface.comHandle23-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 24:hres = txnHandle->comInterface.comHandle24-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 25:hres = txnHandle->comInterface.comHandle25-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
        case 26:hres = txnHandle->comInterface.comHandle26-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;

```



```

case 27:hres = txnHandle->comInterface.comHandle27-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 28:hres = txnHandle->comInterface.comHandle28-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 29:hres = txnHandle->comInterface.comHandle29-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 30:hres = txnHandle->comInterface.comHandle30-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 31:hres = txnHandle->comInterface.comHandle31-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 32:hres = txnHandle->comInterface.comHandle32-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 33:hres = txnHandle->comInterface.comHandle33-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 34:hres = txnHandle->comInterface.comHandle34-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 35:hres = txnHandle->comInterface.comHandle35-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 36:hres = txnHandle->comInterface.comHandle36-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 37:hres = txnHandle->comInterface.comHandle37-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 38:hres = txnHandle->comInterface.comHandle38-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 39:hres = txnHandle->comInterface.comHandle39-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 40:hres = txnHandle->comInterface.comHandle40-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 41:hres = txnHandle->comInterface.comHandle41-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 42:hres = txnHandle->comInterface.comHandle42-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 43:hres = txnHandle->comInterface.comHandle43-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 44:hres = txnHandle->comInterface.comHandle44-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;

```

```

case 45:hres = txnHandle->comInterface.comHandle45-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 46:hres = txnHandle->comInterface.comHandle46-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 47:hres = txnHandle->comInterface.comHandle47-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 48:hres = txnHandle->comInterface.comHandle48-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 49:hres = txnHandle->comInterface.comHandle49-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break;
case 50:hres = txnHandle->comInterface.comHandle50-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);break; }
#else
hres = txnHandle->comInterface.comHandle0-
>doStockLevel(&txnHandle-
>comInterface.size,(UCHAR**)&txnHandle-
>comInterface.txnBuffer);
#endif
}
catch(...)
{
#ifdef PARTITION
html+=sprintf(html,"ERROR : Com Stock call caused
exception, warehouse: %d, COM server %d, ratio
%d.</PRE></BODY></HTML>",stok->in_stok.s_W_ID,txnHandle-
>part_server,ratio);
#else
html+=sprintf(html,"ERROR : Com Stock call caused
exception, warehouse: %d</PRE></BODY></HTML>",stok-
>in_stok.s_W_ID);
#endif
return OK;
}

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

try
{
#ifdef PARTITION
switch (txnHandle->part_server) {
case 0: hres = txnHandle->comInterface.comHandle0-
>doSetComplete();break;
case 1: hres = txnHandle->comInterface.comHandle1-
>doSetComplete();break;
case 2: hres = txnHandle->comInterface.comHandle2-
>doSetComplete();break;
case 3: hres = txnHandle->comInterface.comHandle3-
>doSetComplete();break;
case 4: hres = txnHandle->comInterface.comHandle4-
>doSetComplete();break;
case 5: hres = txnHandle->comInterface.comHandle5-
>doSetComplete();break;

```

```

case 6: hres = txnHandle->comInterface.comHandle6-
>doSetComplete();break;
case 7: hres = txnHandle->comInterface.comHandle7-
>doSetComplete();break;
case 8: hres = txnHandle->comInterface.comHandle8-
>doSetComplete();break;
case 9: hres = txnHandle->comInterface.comHandle9-
>doSetComplete();break;
case 10: hres = txnHandle->comInterface.comHandle10-
>doSetComplete();break;
case 11: hres = txnHandle->comInterface.comHandle11-
>doSetComplete();break;
case 12: hres = txnHandle->comInterface.comHandle12-
>doSetComplete();break;
case 13: hres = txnHandle->comInterface.comHandle13-
>doSetComplete();break;
case 14: hres = txnHandle->comInterface.comHandle14-
>doSetComplete();break;
case 15: hres = txnHandle->comInterface.comHandle15-
>doSetComplete();break;
case 16: hres = txnHandle->comInterface.comHandle16-
>doSetComplete();break;
case 17: hres = txnHandle->comInterface.comHandle17-
>doSetComplete();break;
case 18: hres = txnHandle->comInterface.comHandle18-
>doSetComplete();break;
case 19: hres = txnHandle->comInterface.comHandle19-
>doSetComplete();break;
case 20: hres = txnHandle->comInterface.comHandle20-
>doSetComplete();break;
case 21: hres = txnHandle->comInterface.comHandle21-
>doSetComplete();break;
case 22: hres = txnHandle->comInterface.comHandle22-
>doSetComplete();break;
case 23: hres = txnHandle->comInterface.comHandle23-
>doSetComplete();break;
case 24: hres = txnHandle->comInterface.comHandle24-
>doSetComplete();break;
case 25: hres = txnHandle->comInterface.comHandle25-
>doSetComplete();break;
case 26: hres = txnHandle->comInterface.comHandle26-
>doSetComplete();break;
case 27: hres = txnHandle->comInterface.comHandle27-
>doSetComplete();break;
case 28: hres = txnHandle->comInterface.comHandle28-
>doSetComplete();break;
case 29: hres = txnHandle->comInterface.comHandle29-
>doSetComplete();break;
case 30: hres = txnHandle->comInterface.comHandle30-
>doSetComplete();break;
case 31: hres = txnHandle->comInterface.comHandle31-
>doSetComplete();break;
case 32: hres = txnHandle->comInterface.comHandle32-
>doSetComplete();break;
case 33: hres = txnHandle->comInterface.comHandle33-
>doSetComplete();break;
case 34: hres = txnHandle->comInterface.comHandle34-
>doSetComplete();break;
case 35: hres = txnHandle->comInterface.comHandle35-
>doSetComplete();break;
case 36: hres = txnHandle->comInterface.comHandle36-
>doSetComplete();break;
case 37: hres = txnHandle->comInterface.comHandle37-
>doSetComplete();break;
case 38: hres = txnHandle->comInterface.comHandle38-
>doSetComplete();break;
case 39: hres = txnHandle->comInterface.comHandle39-
>doSetComplete();break;
case 40: hres = txnHandle->comInterface.comHandle40-
>doSetComplete();break;
case 41: hres = txnHandle->comInterface.comHandle41-
>doSetComplete();break;

```

```

case 42: hres = txnHandle->comInterface.comHandle42-
>doSetComplete();break;
case 43: hres = txnHandle->comInterface.comHandle43-
>doSetComplete();break;
case 44: hres = txnHandle->comInterface.comHandle44-
>doSetComplete();break;
case 45: hres = txnHandle->comInterface.comHandle45-
>doSetComplete();break;
case 46: hres = txnHandle->comInterface.comHandle46-
>doSetComplete();break;
case 47: hres = txnHandle->comInterface.comHandle47-
>doSetComplete();break;
case 48: hres = txnHandle->comInterface.comHandle48-
>doSetComplete();break;
case 49: hres = txnHandle->comInterface.comHandle49-
>doSetComplete();break;
case 50: hres = txnHandle->comInterface.comHandle50-
>doSetComplete();break;
}
#else
hres = txnHandle->comInterface.comHandle0->doSetComplete();
#endif
}
catch(...)
{
    ERRORMSG("txnHandle address:"<<hex<<txnHandle<<
"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_DLKV_QUEUE_FULL:
            appendText(&html,"Execution Status: ERROR: Rollback DLKV
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;
    FILE *server_logtrans;

    DEBUGMSG("dlvyThread " << GetCurrentThreadId() << " entered
dlvyThreadEntry, calling db_connect to db:" << dbName << endl);

    void *connectHandle;
    //connect to database.
    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);
    return;
}

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,dlvyThreadID);
dlvyLog = fopen(logFileName,"w");
if(!dlvyLog)
{
    ERRORMSG("dlvyThread " << GetCurrentThreadId() <<"
unable to open dlvy log "
<< dlvyLogPath <<"\\del_" << dlvyThreadID << endl);
    DEBUGMSG("dlvyThread " << GetCurrentThreadId() <<"
unable to open 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,INFINITE);

    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_dlv.s_O_CARRIER_ID =
dlvyQueueData.in_s_0_CARRIER_ID;
        dlvyTxn.in_dlv.s_W_ID = dlvyQueueData.warehouse;
        dlvyTxn.out_dlv.s_transtatus = -1;

        //increment dlvy count
        dlvyCount++;

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

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

        _ftime(&endProcessTime);

        rc = dlvyTxn.out_dlv.s_transtatus;

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

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

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

                buffer+=bytesWritten;
            }
        }
        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
%d. Thread exiting.\n",GetCurrentThreadId());
        fflush(dlvyLog);

        LeaveCriticalSection(&dlvyQueueLock);
    }
} //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 unserved
queued dlvy txn."<<endl);
            DEBUGMSG("Error dlvy queue inserting over unserved
queued dlvy txn."<<endl);
            LeaveCriticalSection(&dlvyQueueLock);
            return ERR_DLVS_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()](&commandBl
ock,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;
#ifdef PARTITION
    txnHandle->part_server = termArray[terminal].part_serv;
#endif
    DEBUGMSG("Users terminal:"<<terminal<<", stored
warehouse:"<<txnHandle->w_id<<
" district:"<<txnHandle->d_id<<" terminalID
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;
#ifdef PARTITION
    termArray[termNextFree].part_serv = (txnHandle-
>w_id%(numWarehouse))/ratio;
#endif

    //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.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

```

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  :
**
*****
*/

```

```

#ifdef __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 <orapcc.h>
#endif

```

```

#include <comsvcs.h>

```

```

//
// Terminal struct
//
struct TERM_ENTRY
{
    int    terminalID;
    bool   terminalInUse;
#ifdef PARTITION
    int    part_serv;
#endif
    int    w_id;
    short  d_id;
};

```

```

//
// COM interface
//
struct COM_HANDLE
{

```

```

    Itpcc_com0 *comHandle0;
#ifdef PARTITION
    Itpcc_com1 *comHandle1;
    Itpcc_com2 *comHandle2;
    Itpcc_com3 *comHandle3;
    Itpcc_com4 *comHandle4;
    Itpcc_com5 *comHandle5;
    Itpcc_com6 *comHandle6;
    Itpcc_com7 *comHandle7;
    Itpcc_com8 *comHandle8;
    Itpcc_com9 *comHandle9;
    Itpcc_com10 *comHandle10;
    Itpcc_com11 *comHandle11;
    Itpcc_com12 *comHandle12;
    Itpcc_com13 *comHandle13;
    Itpcc_com14 *comHandle14;
    Itpcc_com15 *comHandle15;
    Itpcc_com16 *comHandle16;
    Itpcc_com17 *comHandle17;
    Itpcc_com18 *comHandle18;
    Itpcc_com19 *comHandle19;
    Itpcc_com20 *comHandle20;
    Itpcc_com21 *comHandle21;

```

```

ltpcc_com22 *comHandle22;
ltpcc_com23 *comHandle23;
ltpcc_com24 *comHandle24;
ltpcc_com25 *comHandle25;
ltpcc_com26 *comHandle26;
ltpcc_com27 *comHandle27;
ltpcc_com28 *comHandle28;
ltpcc_com29 *comHandle29;
ltpcc_com30 *comHandle30;
ltpcc_com31 *comHandle31;
ltpcc_com32 *comHandle32;
ltpcc_com33 *comHandle33;
ltpcc_com34 *comHandle34;
ltpcc_com35 *comHandle35;
ltpcc_com36 *comHandle36;
ltpcc_com37 *comHandle37;
ltpcc_com38 *comHandle38;
ltpcc_com39 *comHandle39;
ltpcc_com40 *comHandle40;
ltpcc_com41 *comHandle41;
ltpcc_com42 *comHandle42;
ltpcc_com43 *comHandle43;
ltpcc_com44 *comHandle44;
ltpcc_com45 *comHandle45;
ltpcc_com46 *comHandle46;
ltpcc_com47 *comHandle47;
ltpcc_com48 *comHandle48;
ltpcc_com49 *comHandle49;
ltpcc_com50 *comHandle50;
#endif
char *txnBuffer;
int size;
};

//////////
// TXN handle
//////////
struct TXN_HANDLE
{
#ifdef PARTITION
int part_server;
#endif
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 readRegistryValues();
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

```

A.2 Transaction Code

pldel.cpp

```

#include "../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"

extern int sqlfile(FILE* server_logtrans,char *fnam, text *linebuf);
extern void oralogprintf(FILE* server_logtrans,char *format, ...);

```

```

extern char OracleHome[256];

#define DMLRETDEL

#define SQLTXT "BEGIN initpcc.init_del ; END;"

#define SQLTXT1 "DELETE FROM nord WHERE no_d_id = :d_id \
    AND no_w_id = :w_id and rownum <= 1 \
    RETURNING no_o_id into :o_id "

#define SQLTXT3 "UPDATE ordr SET o_carrier_id = :carrier_id \
    WHERE o_id = :o_id and o_d_id = :d_id and o_w_id \
    = :w_id \
    returning o_c_id into :o_c_id"

#define SQLTXT4 "UPDATE ordl \
    SET ol_delivery_d = :cr_date \
    WHERE ol_w_id = :w_id AND ol_d_id = :d_id AND ol_o_id = :o_id \
    RETURNING sum(ol_amount) into :ol_amount "

#define SQLTXT6 "UPDATE cust SET c_balance = c_balance \
+ :amt, \
    c_delivery_cnt = c_delivery_cnt + 1 WHERE c_w_id = :w_id AND \
    c_d_id = :d_id AND c_id = :c_id"

#define NDISTS 10
#define ROWIDLEN 20

struct delctx {
    sb2 del_o_id_ind[NDISTS];
    sb2 d_id_ind[NDISTS];
    sb2 c_id_ind[NDISTS];
    sb2 del_date_ind[NDISTS];
    sb2 carrier_id_ind[NDISTS];
    sb2 amt_ind[NDISTS];

    ub4 del_o_id_len[NDISTS];
    ub4 c_id_len[NDISTS];
    int oid_ctx;
    int cid_ctx;
    OCIBind *olamt_bp;

    ub2 w_id_len[NDISTS];
    ub2 d_id_len[NDISTS];
    ub2 del_date_len[NDISTS];
    ub2 carrier_id_len[NDISTS];
    ub2 amt_len[NDISTS];

    ub2 del_o_id_rcode[NDISTS];
    ub2 cons_rcode[NDISTS];
    ub2 w_id_rcode[NDISTS];
    ub2 d_id_rcode[NDISTS];
    ub2 c_id_rcode[NDISTS];
    ub2 del_date_rcode[NDISTS];
    ub2 carrier_id_rcode[NDISTS];
    ub2 amt_rcode[NDISTS];

    int del_o_id[NDISTS];
    int del_d_id[NDISTS];
    int cons[NDISTS];
    int w_id[NDISTS];
    int d_id[NDISTS];
    int c_id[NDISTS];
    int carrier_id[NDISTS];
    int amt[NDISTS];
    ub4 del_o_id_rcnt;
    int retry;
    OCIRowid *no_rowid_ptr[NDISTS];
    OCIRowid *o_rowid_ptr[NDISTS];
    OCIDate del_date[NDISTS];

```

```

OCISmt *curd0;
OCISmt *curd1;
OCISmt *curd2;
OCISmt *curd3;
OCISmt *curd4;
OCISmt *curd5;
OCISmt *curd6;
OCISmt *curdtest;

OCIBind *w_id_bp;
OCIBind *w_id_bp3;
OCIBind *w_id_bp4;
OCIBind *w_id_bp5;
OCIBind *w_id_bp6;
OCIBind *d_id_bp;
OCIBind *d_id_bp3;
OCIBind *d_id_bp4;
OCIBind *d_id_bp6;
OCIBind *o_id_bp;
OCIBind *cr_date_bp;
OCIBind *c_id_bp;
OCIBind *c_id_bp3;
OCIBind *no_rowid_bp;
OCIBind *carrier_id_bp;
OCIBind *o_rowid_bp;
OCIBind *del_o_id_bp;
OCIBind *del_o_id_bp3;
OCIBind *amt_bp;
OCIBind *bstr1_bp[10];
OCIBind *bstr2_bp[10];
OCIBind *retry_bp;
OCIDefine *inum_dp;
OCIDefine *d_id_dp;
OCIDefine *del_o_id_dp;
OCIDefine *no_rowid_dp;
OCIDefine *c_id_dp;
OCIDefine *o_rowid_dp;
OCIDefine *cons_dp;
OCIDefine *amt_dp;

int norow;
};

typedef struct delctx delctx;
struct pldelctx {

    ub2 del_d_id_len[NDISTS];
    ub2 del_o_id_len[NDISTS];

    ub2 w_id_len;
    ub2 d_id_len[NDISTS];
    ub2 o_c_id_len[NDISTS];
    ub2 sums_len[NDISTS];
    ub2 carrier_id_len;
    ub2 ordcnt_len;
    ub2 del_date_len;

    int del_o_id[NDISTS];
    int del_d_id[NDISTS];
    int o_c_id[NDISTS];
#ifdef USE_IEEE_NUMBER
    float sums[NDISTS];
#else
    int sums[NDISTS];
#endif
    OCIDate del_date;
    int carrier_id;
    int ordcnt;

    ub4 del_o_id_rcnt;
    ub4 del_d_id_rcnt;

```

```

ub4 o_c_id_rcnt;
ub4 sums_rcnt;

int retry;
OCIStmt *curp1;
OCIStmt *curp2;
OCIBind *w_id_bp;
OCIBind *d_id_bp;
OCIBind *o_id_bp;
OCIBind *o_c_id_bp;
OCIBind *ordcnt_bp;
OCIBind *sums_bp;
OCIBind *del_date_bp;
OCIBind *carrier_id_bp;
OCIBind *retry_bp;

int norow;

};
typedef struct pldelctx pldelctx;

#ifdef DMLRETDEL
struct amtctx {
    int ol_amt[NITEMS];
    sb2 ol_amt_ind[NITEMS];
    ub4 ol_amt_len[NITEMS];
    ub2 ol_amt_rcode[NITEMS];
    int ol_cnt;
};
typedef struct amtctx amtctx;

#endif

tkvcinit (ora_cn_data_t *ora_SlotDataP)
{
    char sqlfilename[256];
    text stmbuff[SQL_BUF_SIZE];
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;

    delctx *dctx;
    pldelctx *pldctx;
    dlvy_wrapper *delP;
    OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCIStmt *curi = ora_SlotDataP->curi;

    dctx = (delctx *) malloc (sizeof(delctx));
    memset(dctx,(char)0,sizeof(delctx));
    dctx->norow = 0;

    ora_SlotDataP->dctx = (void *)dctx;
    delP = &ora_SlotDataP->delP;

    pldctx = (pldelctx *) malloc (sizeof(pldelctx));
    DISCARD memset(pldctx,(char)0,(ub4)sizeof(pldelctx));
    ora_SlotDataP->pldctx = (void *)pldctx;
    /* Initialize */
    DISCARD OCIHandleAlloc(tpcenv, (dvoid**)&pldctx->curp1,
OCI_HTYPE_STMT, 0,
(dvoid**)0);
    DISCARD sprintf ((char *) stmbuff, SQLTXT);
    DISCARD OCIStmtPrepare(pldctx->curp1, (OCIError *)errhp,
stmbuff,
(ub4) strlen((char *)stmbuff),
OCI_NTV_SYNTAX, OCI_DEFAULT);
    DISCARD OCIERROR(errhp,
OCIStmtExecute(tpcsvc,pldctx->curp1,(OCIError
*)errhp,1,0,NULLP(OCISnapshot),

```

```

NULLP(OCISnapshot), OCI_DEFAULT));

    DISCARD OCIHandleAlloc(tpcenv,(dvoid**) &pldctx->curp2,
OCI_HTYPE_STMT,
0, (dvoid**)0);
    #if defined(ISO5) || defined(ISO6) || defined(ISO8)
    #if defined(ISO5)
        sprintf(sqlfilename,"%s%s",OracleHome,"/tpcc-
kit/benchrun/blocks/tkvcpdel_iso5.sql");
        sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuff);
    #endif
    #if defined(ISO6)
        sprintf(sqlfilename,"%s%s",OracleHome,"/tpcc-
kit/benchrun/blocks/tkvcpdel_iso6.sql");
        sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuff);
    #endif
    #if defined(ISO8)
        sprintf(sqlfilename,"%s%s",OracleHome,"/tpcc-
kit/benchrun/blocks/tkvcpdel_iso8.sql");
        sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuff);
    #endif
    #else
        sprintf(sqlfilename,"%s%s",OracleHome,"/tpcc-
kit/benchrun/blocks/tkvcpdel.sql");
        sqlfile(ora_SlotDataP->server_logtrans,sqlfilename,stmbuff);
    #endif
    DISCARD OCIStmtPrepare(pldctx->curp2, (OCIError *)errhp,
stmbuff,
(ub4)strlen((char *)stmbuff), OCI_NTV_SYNTAX,
OCI_DEFAULT);
    OCIBNDPL(pldctx->curp2, pldctx->w_id_bp , (OCIError
*)errhp,":w_id",
ADR(delP->in_dlvy.s_W_ID), SIZ(int), SQLT_INT,&pldctx-
>w_id_len);
    OCIBNDPL(pldctx->curp2, pldctx->ordcnt_bp , (OCIError
*)errhp,":ordcnt",
ADR(pldctx->ordcnt), SIZ(int), SQLT_INT,&pldctx-
>ordcnt_len);

    OCIBNDPL(pldctx->curp2, pldctx->del_date_bp,(OCIError
*)errhp,":now",
ADR(pldctx->del_date), SIZ(OCIDate),SQLT_ODT,&pldctx-
>del_date_len);

    OCIBNDPL(pldctx->curp2, pldctx->carrier_id_bp , (OCIError
*)errhp,
":carrier_id", ADR(delP->in_dlvy.s_O_CARRIER_ID), SIZ(int),
SQLT_INT, &pldctx->carrier_id_len);

    OCIBNDPLA(pldctx->curp2, pldctx->d_id_bp, (OCIError
*)errhp,":d_id",
pldctx->del_d_id, SIZ(int),SQLT_INT, pldctx-
>del_d_id_len,
NDISTS, &pldctx->del_d_id_rcnt);
    OCIBNDPLA(pldctx->curp2, pldctx->o_id_bp, (OCIError
*)errhp,":order_id",
pldctx->del_o_id,SIZ(int),SQLT_INT, pldctx-
>del_o_id_len,NDISTS,
&pldctx->del_o_id_rcnt);
    #ifdef USE_IEEE_NUMBER
    OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, (OCIError
*)errhp,":sums",
pldctx->sums,SIZ(float),SQLT_BFLOAT, pldctx-
>sums_len,NDISTS,
&pldctx->sums_rcnt);
    #else
    OCIBNDPLA(pldctx->curp2, pldctx->sums_bp, (OCIError
*)errhp,":sums",
pldctx->sums,SIZ(int),SQLT_INT, pldctx-
>sums_len,NDISTS,
&pldctx->sums_rcnt);
    #endif

```

```

#endif
OCIBNDPLA(pldctx->curp2, pldctx->o_c_id_bp, (OCLError
*)errhp,"o_c_id",
    pldctx->o_c_id,SIZ(int),SQLT_INT, pldctx-
>o_c_id_len,NDISTS,
    &pldctx->o_c_id_rcnt);
OCIBND(pldctx->curp2, pldctx->retry_bp, (OCLError
*)errhp,"retry",
    ADR(pldctx->retry), SIZ(int),SQLT_INT);

return (0);
}

tkvcd (ora_cn_data_t *ora_SlotDataP)
{
    ub4 i;
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;

    delctx *dctx = (delctx *)ora_SlotDataP->dctx;
    pldelctx *pldctx = (pldelctx *)ora_SlotDataP->pldctx;
#ifdef DMLRETDEL /* VMM 1/13/98 */
    amtctx *actx = (amtctx *)ora_SlotDataP->actx;
#endif /* DMLRETDEL */
    dlvy_wrapper *delP = &ora_SlotDataP->delP;
    OCIEEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCISstmt *curi = ora_SlotDataP->curi;

retry:
    pldctx->w_id_len = sizeof(int);
    pldctx->carrier_id_len = sizeof(int);
    for (i = 0; i < NDISTS; i++)
    {
        pldctx->del_o_id_len[i] = sizeof(int);
    }
    pldctx->del_date_len = DEL_DATE_LEN;
    DISCARD memcpy(&pldctx->del_date,&delP-
>in_dlvy.cr_date,sizeof(OCIDate));

    pldctx->retry=0;

    delP->out_dlvy.error = OCISstmtExecute(tpcsvc,pldctx-
>curp2,(OCLError *)errhp,1,0,NULLP(CONST OCISnapshot),
        NULLP(OCISnapshot),OCI_DEFAULT);
    if((delP->out_dlvy.error != OCI_SUCCESS) && (delP-
>out_dlvy.error != OCI_NO_DATA))
    {
        DISCARD OCITransRollback(tpcsvc,(OCLError
*)errhp,OCI_DEFAULT);
        delP->out_dlvy.error = OCIEERROR((OCLError *)errhp,delP-
>out_dlvy.error);
        if(delP->out_dlvy.error == NOT_SERIALIZABLE)
        {
            delP->out_dlvy.retry++;
            goto retry;
        }
        else if (delP->out_dlvy.error == RECOVERR)
        {
            delP->out_dlvy.retry++;
            goto retry;
        }
        else if (delP->out_dlvy.error == SNAPSHOT_TOO_OLD)
        {
            delP->out_dlvy.retry++;
            goto retry;
        }
        else
        {

```

```

            return -1;
        }
    }
    for (i = 0; i < NDISTS; i++)
    {
        delP->out_dlvy.s_O_ID[i] = 0;
    }
    for (i = 0; i < pldctx->del_o_id_rcnt; i++)
        delP->out_dlvy.s_O_ID[pldctx->del_d_id[i] - 1] = pldctx-
>del_o_id[i];
    return (0);
}

```

```

void tkvcdone (ora_cn_data_t *ora_SlotDataP)
{
    pldelctx *pldctx = (pldelctx *)ora_SlotDataP->pldctx;

    if (pldctx)
    {
        DISCARD OCIFree((dvoid *)pldctx-
>curp1,OCI_HTYPE_STMT);
        DISCARD OCIFree((dvoid *)pldctx-
>curp2,OCI_HTYPE_STMT);
        DISCARD free(pldctx);
    }
}

```

plnew.cpp

```

#include "../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"

#define SQLTXT2 "BEGIN inittpcc.init_no(:idx1arr); END;"

#define NITEMS 15
#define ROWIDLEN 20
#define OCIROWLEN 20

extern void oralogprintf(FILE *server_logtrans,char *format, ...);

extern char OracleHome[256];
extern int sqlfile(FILE *server_logtrans,char *fnam, text *linebuf);

struct newctx {

    ub2 nol_i_id_len[NITEMS];
    ub2 nol_supply_w_id_len[NITEMS];
    ub2 nol_quantity_len[NITEMS];
    ub2 nol_amount_len[NITEMS];
    ub2 s_quantity_len[NITEMS];
    ub2 i_name_len[NITEMS];
    ub2 i_price_len[NITEMS];
    ub2 s_dist_info_len[NITEMS];
    ub2 ol_o_id_len[NITEMS];
    ub2 ol_number_len[NITEMS];
    ub2 s_remote_len[NITEMS];
    ub2 s_quant_len[NITEMS];
    ub2 ol_dist_info_len[NITEMS];
    ub2 s_bg_len[NITEMS];
#ifdef USE_IEEE_NUMBER
    float ol_quantity[15];
    float ol_amount[15];
    float s_remote[NITEMS];
#else
    int ol_quantity[15];
    int ol_amount[15];
    int s_remote[NITEMS];

```

```

#endif /* USE_IEEE_NUMBER */
int ol_number[15];
char s_dist_info[NITEMS][25];
OCIStmt *curn1;
OCIBind *ol_i_id_bp;
OCIBind *ol_supply_w_id_bp;
OCIBind *i_price_bp;
OCIBind *i_name_bp;
OCIBind *s_bg_bp;
ub4 nol_i_count;
ub4 nol_s_count;
ub4 nol_q_count;
ub4 nol_item_count;
ub4 nol_name_count;
ub4 nol_qty_count;
ub4 nol_bg_count;
ub4 nol_am_count;
ub4 s_remote_count;
OCIStmt *curn2;
OCIBind *ol_quantity_bp;
OCIBind *s_remote_bp;
OCIBind *s_quantity_bp;
OCIBind *w_id_bp;
OCIBind *d_id_bp;
OCIBind *c_id_bp;
OCIBind *o_all_local_bp;
OCIBind *o_all_cnt_bp;
OCIBind *w_tax_bp;
OCIBind *d_tax_bp;
OCIBind *o_id_bp;
OCIBind *c_discount_bp;
OCIBind *c_credit_bp;
OCIBind *c_last_bp;
OCIBind *retries_bp;
OCIBind *cr_date_bp;
OCIBind *ol_o_id_bp;
OCIBind *ol_amount_bp;

ub2 w_id_len; /* RP - may need to be (ub2), was (sb2) */
ub2 d_id_len;
ub2 c_id_len;
ub2 o_all_local_len;
ub2 o_ol_cnt_len;
ub2 w_tax_len;
ub2 d_tax_len;
ub2 o_id_len;
ub2 c_discount_len;
ub2 c_credit_len;
ub2 c_last_len;
ub2 retries_len;
ub2 cr_date_len;
};

typedef struct newctx newctx;

tkvcninit (ora_cn_data_t *ora_SlotDataP)
{
    char sqlfilename[256];
    text stmbuff[32*1024];
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;

    newctx *nctx;
    OCIEncv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCIStmt *curi = ora_SlotDataP->curi;
    nord_wrapper *newP;

    oralogprintf(ora_SlotDataP->server_logtrans, ">>tkvcninit\n");
    nctx = (newctx *) malloc (sizeof(newctx));

```

```

DISCARD memset(nctx, (char)0, sizeof(newctx));
ora_SlotDataP->nctx = (void *)nctx;

memset(&ora_SlotDataP->globals, (char)0, sizeof(nord_wrapper));
newP = &ora_SlotDataP->globals;

nctx->w_id_len = sizeof(newP->in_nord.s_W_ID);
nctx->d_id_len = sizeof(newP->in_nord.s_D_ID);
nctx->c_id_len = sizeof(newP->in_nord.s_C_ID);
nctx->o_all_local_len = sizeof(newP->in_nord.s_all_local);
nctx->o_ol_cnt_len = sizeof(newP->out_nord.s_O_OL_CNT);
nctx->w_tax_len = 0;
nctx->d_tax_len = 0;
nctx->o_id_len = sizeof(newP->out_nord.s_O_ID);
nctx->c_discount_len = 0;
nctx->c_credit_len = 0;
nctx->c_last_len = 0;
nctx->retries_len = sizeof(newP->out_nord.retry);
nctx->cr_date_len = sizeof(newP->in_nord.cr_date);

/* open first cursor */
DISCARD OCIERROR((OCIError
*errhp, OCIHandleAlloc(tpcenv, (dvoid **)&nctx->curn1,
OCI_HTYPE_STMT, 0, (dvoid**)0));

#if defined(ISO)
    sprintf(sqlfilename, "%s/tpcc-
kit/benchrun/blocks/tkvcpnw_iso.sql", OracleHome);
    sqlfile(ora_SlotDataP->server_logtrans, sqlfilename, stmbuff);
#else
    #if defined(ISO7)
        sprintf(sqlfilename, "%s/tpcc-
kit/benchrun/blocks/tkvcpnw_iso7.sql", OracleHome);
        sqlfile(ora_SlotDataP->server_logtrans, sqlfilename, stmbuff);
    #else
        sprintf(sqlfilename, "%s/tpcc-
kit/benchrun/blocks/tkvcpnw.sql", OracleHome);
        sqlfile(ora_SlotDataP->server_logtrans, sqlfilename, stmbuff);
    #endif
#endif

DISCARD OCIERROR((OCIError *)errhp, OCIStmtPrepare(nctx-
>curn1, (OCIError *)errhp, stmbuff,
strlen((char *)stmbuff), OCI_NTV_SYNTAX, OCI_DEFAULT));

/* bind variables */
oralogprintf(ora_SlotDataP->server_logtrans, "--tkvcninit\n");

OCIBNDPL(nctx->curn1, nctx->w_id_bp, (OCIError *)errhp,
":w_id", ADR(newP->in_nord.s_W_ID), SIZ(newP->in_nord.s_W_ID),
SQLT_INT, &nctx->w_id_len);
OCIBNDPL(nctx->curn1, nctx->d_id_bp, (OCIError *)errhp,
":d_id", ADR(newP->in_nord.s_D_ID), SIZ(newP->in_nord.s_D_ID),
SQLT_INT, &nctx->d_id_len);
OCIBNDPL(nctx->curn1, nctx->c_id_bp, (OCIError *)errhp,
":c_id", ADR(newP->in_nord.s_C_ID), SIZ(newP->in_nord.s_C_ID),
SQLT_INT, &nctx->c_id_len);
OCIBNDPL(nctx->curn1, nctx->o_all_local_bp, (OCIError *)errhp,
":o_all_local",
ADR(newP->in_nord.s_all_local), SIZ(newP-
>in_nord.s_all_local), SQLT_INT, &nctx->o_all_local_len);
OCIBNDPL(nctx->curn1, nctx->o_all_cnt_bp, (OCIError *)errhp,
":o_ol_cnt", ADR(newP->out_nord.s_O_OL_CNT),
SIZ(newP->out_nord.s_O_OL_CNT), SQLT_INT, &nctx-
>o_ol_cnt_len);
OCIBNDPL(nctx->curn1, nctx->w_tax_bp, (OCIError *)errhp,
":w_tax", ADR(newP->out_nord.s_W_TAX), SIZ(newP-
>out_nord.s_W_TAX),
SQLT_FLT, &nctx->w_tax_len);
OCIBNDPL(nctx->curn1, nctx->d_tax_bp, (OCIError *)errhp,
":d_tax", ADR(newP->out_nord.s_D_TAX), SIZ(newP-
>out_nord.s_D_TAX),
SQLT_FLT, &nctx->d_tax_len);

```

```

    OCIBNDPL(nctx->curn1, nctx->o_id_bp, (OCIErr *errhp,
"o_id",ADR(newP->out_nord.s_O_ID),SIZ(newP-
>out_nord.s_O_ID),
    SQT_INT, &nctx->o_id_len);
    OCIBNDPL(nctx->curn1, nctx->c_discount_bp, (OCIErr *errhp,
"c_discount",
    ADR(newP->out_nord.s_C_DISCOUNT), SIZ(newP-
>out_nord.s_C_DISCOUNT),SQT_FLT, &nctx->c_discount_len);
    OCIBNDPL(nctx->curn1, nctx->c_credit_bp, (OCIErr *errhp,
"c_credit",newP->out_nord.s_C_CREDIT,
    SIZ(newP->out_nord.s_C_CREDIT),SQT_CHR, &nctx-
>c_credit_len);
    OCIBNDPL(nctx->curn1, nctx->c_last_bp, (OCIErr *errhp,
"c_last",newP->out_nord.s_C_LAST,SIZ(newP-
>out_nord.s_C_LAST),
    SQT_STR, &nctx->c_last_len);
    OCIBNDPL(nctx->curn1, nctx->retries_bp, (OCIErr *errhp,
"retry",ADR(newP->out_nord.retry),
    SIZ(newP->out_nord.retry),SQT_INT, &nctx->retries_len);
    OCIBNDPL(nctx->curn1, nctx->cr_date_bp, (OCIErr *errhp,
"cr_date",&newP->in_nord.cr_date,
    SIZ(OCIDate), SQT_ODT, &nctx->cr_date_len);

    OCIBNDPLA(nctx->curn1, nctx->ol_i_id_bp,(OCIErr
*)errhp,"ol_i_id",newP->in_nord.s_OL_I_ID,
    SIZ(newP->in_nord.s_OL_I_ID[0]), SQT_INT, nctx-
>nol_i_id_len,NITEMS,&nctx->nol_i_count);
    OCIBNDPLA(nctx->curn1, nctx->ol_supply_w_id_bp, (OCIErr
*)errhp,"ol_supply_w_id",
    newP->in_nord.s_OL_SUPPLY_W_ID,SIZ(newP-
>in_nord.s_OL_SUPPLY_W_ID[0]),SQT_INT, nctx-
>nol_supply_w_id_len,
    NITEMS, &nctx->nol_s_count);
#ifdef USE_IEEE_NUMBER
    OCIBNDPLA(nctx->curn1, nctx->ol_quantity_bp,(OCIErr
*)errhp,"ol_quantity",
    newP->in_nord.ol_quantity, SIZ(newP-
>in_nord.ol_quantity[0]),SQT_FLT,nctx->nol_quantity_len,
    NITEMS,&nctx->nol_q_count);

    OCIBNDPLA(nctx->curn1, nctx->i_price_bp,(OCIErr
*)errhp,"i_price",newP->out_nord.s_I_PRICE,SIZ(newP-
>out_nord.s_I_PRICE[0]),
    SQT_FLT, nctx->i_price_len, NITEMS, &nctx-
>nol_item_count);
#else
    OCIBNDPLA(nctx->curn1, nctx->ol_quantity_bp,(OCIErr
*)errhp,"ol_quantity",
    newP->in_nord.s_OL_QUANTITY, SIZ(newP-
>in_nord.s_OL_QUANTITY[0]),SQT_INT,nctx->nol_quantity_len,
    NITEMS,&nctx->nol_q_count);
    OCIBNDPLA(nctx->curn1, nctx->i_price_bp,(OCIErr
*)errhp,"i_price",newP->out_nord.s_I_PRICE,SIZ(newP-
>out_nord.s_I_PRICE[0]),
    SQT_INT, nctx->i_price_len, NITEMS, &nctx-
>nol_item_count);
#endif
    OCIBNDPLA(nctx->curn1, nctx->i_name_bp,(OCIErr
*)errhp,"i_name",newP->out_nord.s_I_NAME,
    SIZ(newP->out_nord.s_I_NAME[0]),SQT_STR, nctx-
>i_name_len,NITEMS,
    &nctx->nol_name_count);
#ifdef USE_IEEE_NUMBER
    OCIBNDPLA(nctx->curn1, nctx->s_quantity_bp,(OCIErr
*)errhp,"s_quantity",newP->out_nord.s_S_QUANTITY,
    SIZ(newP->out_nord.s_S_QUANTITY[0]), SQT_FLT,nctx-
>s_quant_len,NITEMS,&nctx->nol_qty_count);
#else
    OCIBNDPLA(nctx->curn1, nctx->s_quantity_bp,(OCIErr
*)errhp,"s_quantity",newP->out_nord.s_S_QUANTITY,
    SIZ(newP->out_nord.s_S_QUANTITY[0]), SQT_INT,nctx-
>s_quant_len,NITEMS,&nctx->nol_qty_count);

```

```

#endif
    OCIBNDPLA(nctx->curn1, nctx->s_bg_bp,(OCIErr
*)errhp,"brand_generic",newP->out_nord.s_brand_generic,
    SIZ(newP->out_nord.s_brand_generic[0]), SQT_CHR,nctx-
>s_bg_len,NITEMS,&nctx->nol_bg_count);
#ifdef USE_IEEE_NUMBER
    OCIBNDPLA(nctx->curn1, nctx->ol_amount_bp,(OCIErr
*)errhp,"ol_amount",newP->out_nord.s_OL_AMOUNT,
    SIZ(newP->out_nord.s_OL_AMOUNT[0]),SQT_FLT, nctx-
>nol_amount_len,NITEMS,&nctx->nol_am_count);
    OCIBNDPLA(nctx->curn1, nctx->s_remote_bp,(OCIErr
*)errhp,"s_remote",nctx->s_remote,
    SIZ(nctx->s_remote[15]),SQT_FLT, nctx-
>s_remote_len,NITEMS,&nctx->s_remote_count);
#else
    OCIBNDPLA(nctx->curn1, nctx->ol_amount_bp,(OCIErr
*)errhp,"ol_amount",newP->out_nord.s_OL_AMOUNT,
    SIZ(newP->out_nord.s_OL_AMOUNT[0]),SQT_INT, nctx-
>nol_amount_len,NITEMS,&nctx->nol_am_count);
    OCIBNDPLA(nctx->curn1, nctx->s_remote_bp,(OCIErr
*)errhp,"s_remote",nctx->s_remote,
    SIZ(nctx->s_remote[0]),SQT_INT, nctx-
>s_remote_len,NITEMS,&nctx->s_remote_count);
#endif

```

```

/* open second cursor */
DISCARD OCIERROR((OCIErr *)errhp,OCIHandleAlloc(tpcenv,
(dvoid **)(&nctx->curn2),
    OCI_HTYPE_STMT, 0, (dvoid**)0));
DISCARD sprintf ((char *) stmbuf, SQTXT2);
DISCARD OCIERROR((OCIErr *)errhp,OCIStmtPrepare(nctx-
>curn2, (OCIErr *)errhp, stmbuf,
    strlen((char *)stmbuf), OCI_NTV_SYNTAX,
    OCI_DEFAULT));

```

```

/* execute second cursor to init newinit package */

```

```

{
    int idx1arr[NITEMS];
    OCIBind *idx1arr_bp;
    ub2 idx1arr_len[NITEMS];
    sb2 idx1arr_ind[NITEMS];
    ub4 idx1arr_count;
    ub2 idx;

    for (idx = 0; idx < NITEMS; idx++) {
        idx1arr[idx] = idx + 1;
        idx1arr_ind[idx] = TRUE;
        idx1arr_len[idx] = sizeof(int);
    }
    idx1arr_count = NITEMS;
    newP->out_nord.s_O_OL_CNT = NITEMS;
}

```

```

/* Bind array */
OCIBNDPLA(nctx->curn2, idx1arr_bp,(OCIErr
*)errhp,"idx1arr",idx1arr,
    SIZ(int), SQT_INT, idx1arr_len, NITEMS,&idx1arr_count);

```

```

newP->out_nord.terror = OCIStmtExecute(tpcsvc,nctx-
>curn2,(OCIErr *)errhp,1,0,
    NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
if(newP->out_nord.terror != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,(OCIErr *)errhp,OCI_DEFAULT);
    newP->out_nord.terror = OCIERROR((OCIErr *)errhp,newP-
>out_nord.terror);
    return -1;
}
}

```

```

oralogprintf(ora_SlotDataP->server_logtrans,"<<tkvcninit\n");

```

```

return (0);
}

tkvcn (ora_cn_data_t *ora_SlotDataP)
{
    int i;
    int rcount;
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;

    OCIEncv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCIStmt *curi = ora_SlotDataP->curi;
    nord_wrapper *newP = &ora_SlotDataP->globals;
    newctx *nctx = (newctx *)ora_SlotDataP->nctx;

retry:
    newP->out_nord.status = 0;          /* number of invalid
items */

    /* get number of order lines, and check if all are local */

    newP->out_nord.s_O_OL_CNT = NITEMS;
    newP->in_nord.s_all_local = 1;
    for (i = 0; i < NITEMS; i++) {
        if (newP->in_nord.s_OL_I_ID[i] == 0) {
            newP->out_nord.s_O_OL_CNT = i;
            break;
        }
        if (newP->in_nord.s_OL_SUPPLY_W_ID[i] != newP-
>in_nord.s_W_ID) {
#ifdef USE_IEEE_NUMBER
            nctx->s_remote[i] = 1.0;
#else
            nctx->s_remote[i] = 1;
#endif
        }
        newP->in_nord.s_all_local = 0;
    }
    else
        nctx->s_remote[i] = 0;
}

nctx->w_id_len = sizeof(newP->in_nord.s_W_ID);
nctx->d_id_len = sizeof(newP->in_nord.s_D_ID);
nctx->c_id_len = sizeof(newP->in_nord.s_C_ID);
nctx->o_all_local_len = sizeof(newP->in_nord.s_all_local);
nctx->o_ol_cnt_len = sizeof(newP->out_nord.s_O_OL_CNT);
nctx->w_tax_len = 0;
nctx->d_tax_len = 0;
nctx->o_id_len = sizeof(newP->out_nord.s_O_ID);
nctx->c_discount_len = 0;
nctx->c_credit_len = 0;
nctx->c_last_len = 0;
nctx->retries_len = sizeof(newP->out_nord.retry);
nctx->cr_date_len = sizeof(newP->in_nord.cr_date);
/* this is the row count */
rcount = newP->out_nord.s_O_OL_CNT;
nctx->nol_i_count = newP->out_nord.s_O_OL_CNT;
nctx->nol_q_count = newP->out_nord.s_O_OL_CNT;
nctx->nol_s_count = newP->out_nord.s_O_OL_CNT;
nctx->s_remote_count = newP->out_nord.s_O_OL_CNT;

nctx->nol_qty_count = 0;
nctx->nol_bg_count = 0;
nctx->nol_item_count = 0;
nctx->nol_name_count = 0;
nctx->nol_am_count = 0;

```

```

/* initialization for array operations */
for (i = 0; i < newP->out_nord.s_O_OL_CNT; i++) {
    nctx->ol_number[i] = i + 1;
    nctx->nol_i_id_len[i] = sizeof(newP->in_nord.s_OL_I_ID[i]);
    nctx->nol_supply_w_id_len[i] = sizeof(newP-
>in_nord.s_OL_SUPPLY_W_ID[i]);
    nctx->nol_quantity_len[i] = sizeof(newP-
>in_nord.s_OL_QUANTITY[i]);
    nctx->nol_amount_len[i] = sizeof(newP-
>out_nord.s_OL_AMOUNT[i]);
    nctx->ol_o_id_len[i] = sizeof(newP->out_nord.s_O_ID);
    nctx->ol_number_len[i] = sizeof(int);
    nctx->ol_dist_info_len[i] = nctx->s_dist_info_len[i];
    nctx->s_remote_len[i] = sizeof(int);
    nctx->s_quant_len[i] = sizeof(int);
    nctx->i_name_len[i] = 0;
    nctx->s_bg_len[i] = 0;
}
for (i = newP->out_nord.s_O_OL_CNT; i < NITEMS; i++) {
    nctx->nol_i_id_len[i] = 0;
    nctx->nol_supply_w_id_len[i] = 0;
    nctx->nol_quantity_len[i] = 0;
    nctx->nol_amount_len[i] = 0;
    nctx->ol_o_id_len[i] = 0;
    nctx->ol_number_len[i] = 0;
    nctx->ol_dist_info_len[i] = 0;
    nctx->s_remote_len[i] = 0;
    nctx->s_quant_len[i] = 0;
    nctx->i_name_len[i] = 0;
    nctx->s_bg_len[i] = 0;
}

newP->out_nord.error = OCIStmtExecute(tpcsvc,nctx-
>curi,(OCIError *)errhp,1,0,0,0,
OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);

if(newP->out_nord.error != OCI_SUCCESS) {
    OCITransRollback(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
    newP->out_nord.error =
ocierror(server_logtrans, __FILE__, __LINE__, errhp, newP-
>out_nord.error);
    oralogprintf(ora_SlotDataP->server_logtrans, "newP-
>out_nord.error %d\n", newP->out_nord.error);
    if(newP->out_nord.error == NOT_SERIALIZABLE) {
        newP->out_nord.retry++;
        goto retry;
    } else if (newP->out_nord.error == RECOVERR) {
        newP->out_nord.retry++;
        goto retry;
    } else if (newP->out_nord.error == SNAPSHOT_TOO_OLD) {
        newP->out_nord.retry++;
        goto retry;
    } else {
        return -1;
    }
}

/* did the txn succeed ? */
if (rcount != newP->out_nord.s_O_OL_CNT)
{
    newP->out_nord.status = rcount - newP-
>out_nord.s_O_OL_CNT;
    newP->out_nord.s_O_OL_CNT = rcount;
    newP->out_nord.error = 100;
}
newP->out_nord.s_total_amount = 0;
for (i = 0; i < newP->out_nord.s_O_OL_CNT; i++)
{
    newP->out_nord.s_total_amount += newP-
>out_nord.s_OL_AMOUNT[i];
}

```



```

    }
    newP->out_nord.s_total_amount *= ((float)(1.0 - newP-
>out_nord.s_C_DISCOUNT)) * (float)(1.0 + ((float)(newP-
>out_nord.s_D_TAX)) + ((float)(newP->out_nord.s_W_TAX)));
    newP->out_nord.s_total_amount = newP-
>out_nord.s_total_amount;

    return(newP->out_nord.terror);
}

void tkvcndone (ora_cn_data_t *ora_SlotDataP)

{
    newctx *nctx = (newctx *)ora_SlotDataP->nctx;

    if (nctx)
    {
        DISCARD OCIHandleFree((dvoid *)nctx-
>curn1,OCI_HTYPE_STMT);
        DISCARD OCIHandleFree((dvoid *)nctx-
>curn2,OCI_HTYPE_STMT);
        free (nctx);
    }
}

plord.cpp

#include "../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"
#include "tpccflags.h"

extern void oralogprintf(FILE *,char *format, ...);

/*-----
PRIVATE TYPES AND CONSTANTS
-----*/

/*-----
STATIC FUNCTION DECLARATIONS
-----*/

#define SQLCUR0 "SELECT rowid FROM cust \
WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last
= :c_last \
ORDER BY c_last, c_d_id, c_w_id, c_first"

#define SQLCUR1 "SELECT /*+ USE_NL(cust) INDEX_DESC(ordr
iordr2) */ \
c_id, c_balance, c_first, c_middle, c_last, \
o_id, o_entry_d, o_carrier_id, o_ol_cnt, ordr.rowid \
FROM cust, ordr \
WHERE cust.rowid = :cust_rowid \
AND o_d_id = c_d_id AND o_w_id = c_w_id AND
o_c_id = c_id \
ORDER BY o_c_id DESC, o_d_id DESC, o_w_id DESC,
o_id DESC"

#define SQLCUR2 "SELECT /*+ USE_NL(cust) INDEX_DESC (ordr
iordr2) */ \
c_balance, c_first, c_middle, c_last, \
o_id, o_entry_d, o_carrier_id, o_ol_cnt, ordr.rowid \
FROM cust, ordr \
WHERE c_id = :c_id AND c_d_id = :d_id AND c_w_id
= :w_id \

```

```

AND o_d_id = c_d_id AND o_w_id = c_w_id AND o_c_id
= c_id \
ORDER BY o_c_id DESC, o_d_id DESC, o_w_id DESC ,
o_id DESC"

#define SQLCUR3 "SELECT /*+ ORDERED USE_NL(ordl)
CLUSTER(ordl) */ \
ol_i_id, ol_supply_w_id, ol_quantity, ol_amount,
ol_delivery_d \
FROM ordr, ordl \
WHERE ordr.rowid = :ordr_rowid \
AND o_id = ol_o_id AND ol_d_id = o_d_id AND ol_w_id
= o_w_id"

#define SQLCUR4 "SELECT count(c_last) FROM cust \
WHERE c_d_id = :d_id AND c_w_id = :w_id AND c_last
= :c_last "

struct ordctx {

    ub2 c_rowid_len[100];
    ub2 ol_supply_w_id_len[NITEMS];
    ub2 ol_i_id_len[NITEMS];
    ub2 ol_quantity_len[NITEMS];
    ub2 ol_amount_len[NITEMS];
    ub2 ol_delivery_d_len[NITEMS];
    ub2 ol_w_id_len;
    ub2 ol_d_id_len;
    ub2 ol_o_id_len;

    ub4 ol_supply_w_id_csize;
    ub4 ol_i_id_csize;
    ub4 ol_quantity_csize;
    ub4 ol_amount_csize;
    ub4 ol_delivery_d_csize;
    ub4 ol_w_id_csize;
    ub4 ol_d_id_csize;
    ub4 ol_o_id_csize;

    OCISmt *curo0;
    OCISmt *curo1;
    OCISmt *curo2;
    OCISmt *curo3;
    OCISmt *curo4;
    OCIBind *c_id_bp;
    OCIBind *w_id_bp[4];
    OCIBind *d_id_bp[4];
    OCIBind *c_last_bp[2];
    OCIBind *o_id_bp;
    OCIBind *c_rowid_bp;
    OCIBind *o_rowid_bp;
    OCIDefine *c_rowid_dp;
    OCIDefine *c_last_dp[2];
    OCIDefine *c_id_dp;
    OCIDefine *c_first_dp[2];
    OCIDefine *c_middle_dp[2];
    OCIDefine *c_balance_dp[2];
    OCIDefine *o_rowid_dp[2];
    OCIDefine *o_id_dp[2];
    OCIDefine *o_entry_d_dp[2];
    OCIDefine *o_cr_id_dp[2];
    OCIDefine *o_ol_cnt_dp[2];
    OCIDefine *ol_d_d_dp;
    OCIDefine *ol_i_id_dp;
    OCIDefine *ol_supply_w_id_dp;
    OCIDefine *ol_quantity_dp;
    OCIDefine *ol_amount_dp;
    OCIDefine *ol_d_base_dp;
    OCIDefine *c_count_dp;

```

```

OCIRowid *c_rowid_ptr[100];
OCIRowid *c_rowid_cust;
OCIRowid *o_rowid;
int cs;
int cust_idx;
int norow;
int rcount;
int somerows;
};

typedef struct ordctx ordctx;

struct defctx
{
    boolean reexec;
    ub4 count;
};

struct defctx_ordctx
{
    defctx *ctxp;
    ordctx *octx;
};

typedef struct defctx_ordctx defctx_ordctx;

tkvcoint(ora_cn_data_t *ora_SlotDataP)
{
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;

    int i;
    text stmbuf[SQL_BUF_SIZE];

    ordctx *octx;
    defctx *cbctx;
    defctx_ordctx *ctxp_octx;
    ords_wrapper *ordP;
    OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCISmt *curi = ora_SlotDataP->curi;

    octx = (ordctx *) malloc (sizeof(ordctx));
    DISCARD memset(octx,(char)0,sizeof(ordctx));
    octx->cs = 1;
    octx->norow = 0;
    octx->somerows = 10;
    ora_SlotDataP->octx = (void *)octx;

    cbctx = (defctx *) malloc (sizeof(defctx));
    DISCARD memset(cbctx,(char)0,sizeof(defctx));
    ora_SlotDataP->cbctx = (void *)cbctx;

    /* allocate the space */
    ctxp_octx = (defctx_ordctx *) malloc(sizeof(defctx_ordctx));
    ora_SlotDataP->ctxp_octx = (void *)ctxp_octx;

    memset(&ora_SlotDataP->ordP,(char)0,sizeof(ords_wrapper));
    ordP = &ora_SlotDataP->ordP;

    /* get the rowid handles */
    OCIERROR(errhp, OCIDescriptorAlloc((dvoid *)tpcenv,(dvoid
**) &octx->o_rowid,
        (ub4)OCI_DTYPE_ROWID, (size_t) 0, (dvoid **)0));
    for(i=0;i<100;i++) {
        DISCARD OCIERROR(errhp, OCIDescriptorAlloc(tpcenv,
            (dvoid**) &octx->c_rowid_ptr[i],
OCI_DTYPE_ROWID,0,(dvoid**)0));
    }

```

```

// DISCARD OCIERROR(errhp,
// OCIHandleAlloc(tpcenv,(dvoid**) &octx-
>curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**) &octx-
>curo0,OCI_HTYPE_STMT,0,(dvoid**)0));
DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**) &octx-
>curo1,OCI_HTYPE_STMT,0,(dvoid**)0));
DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**) &octx-
>curo2,OCI_HTYPE_STMT,0,(dvoid**)0));
DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**) &octx-
>curo3,OCI_HTYPE_STMT,0,(dvoid**)0));
DISCARD OCIERROR(errhp,
    OCIHandleAlloc(tpcenv,(dvoid**) &octx-
>curo4,OCI_HTYPE_STMT,0,(dvoid**)0));

/* c_id = 0, use find customer by lastname. Get an array of rowid's
back*/
DISCARD sprintf((char *) stmbuf, SQLCUR0);
DISCARD OCIERROR(errhp,
    OCISmtPrepare(octx->curo0,(OCIError
*)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
        OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo0,OCI_HTYPE_STMT,&octx->norow,0,
        OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));
/* get order/customer info back based on rowid */
DISCARD sprintf((char *) stmbuf, SQLCUR1);
DISCARD OCIERROR(errhp,
    OCISmtPrepare(octx->curo1,(OCIError
*)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
        OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo1,OCI_HTYPE_STMT,&octx->norow,0,
        OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));

/* c_id == 0, use lastname to find customer */
DISCARD sprintf((char *) stmbuf, SQLCUR2);
DISCARD OCIERROR(errhp,
    OCISmtPrepare(octx->curo2,(OCIError
*)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
        OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo2,OCI_HTYPE_STMT,&octx->norow,0,
        OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));

DISCARD sprintf((char *) stmbuf, SQLCUR3);
DISCARD OCIERROR(errhp,
    OCISmtPrepare(octx->curo3,(OCIError
*)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
        OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo3,OCI_HTYPE_STMT,&octx->norow,0,
        OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));

DISCARD sprintf((char *) stmbuf, SQLCUR4);
DISCARD OCIERROR(errhp,
    OCISmtPrepare(octx->curo4,(OCIError
*)errhp,stmbuf,(ub4)strlen((char *)stmbuf),
        OCI_NTV_SYNTAX,OCI_DEFAULT));
DISCARD OCIERROR(errhp,
    OCIAttrSet(octx->curo4,OCI_HTYPE_STMT,&octx->norow,0,
        OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));

for (i = 0; i < NITEMS; i++) {
    octx->ol_supply_w_id_len[i] = sizeof(int);

```

```

    octx->ol_i_id_len[i] = sizeof(int);
#ifdef USE_IEEE_NUMBER
    octx->ol_quantity_len[i] = sizeof(float);
    octx->ol_amount_len[i] = sizeof(float);
#else
    octx->ol_quantity_len[i] = sizeof(int);
    octx->ol_amount_len[i] = sizeof(int);
#endif
    octx->ol_delivery_d_len[i] = sizeof(ordP-
>out_ords.s_OL_DELIVERY_D_base[0]);
}
    octx->ol_supply_w_id_csize = NITEMS;
    octx->ol_i_id_csize = NITEMS;
    octx->ol_quantity_csize = NITEMS;
    octx->ol_amount_csize = NITEMS;
    octx->ol_delivery_d_csize = NITEMS;
    octx->ol_w_id_csize = NITEMS;
    octx->ol_o_id_csize = NITEMS;
    octx->ol_d_id_csize = NITEMS;
    octx->ol_w_id_len = sizeof(int);
    octx->ol_d_id_len = sizeof(int);
    octx->ol_o_id_len = sizeof(int);

/* bind variables */

/* c_id (customer id) is not known */
OCIBND(octx->curo0,octx->w_id_bp[0],(OCIErr
*)errhp,"w_id",ADR(ordP->in_ords.s_W_ID),
    SIZ(int),SQLT_INT);
OCIBND(octx->curo0,octx->d_id_bp[0],(OCIErr
*)errhp,"d_id",ADR(ordP->in_ords.s_D_ID),
    SIZ(int),SQLT_INT);
OCIBND(octx->curo0,octx->c_last_bp[0],(OCIErr
*)errhp,"c_last",ordP->in_ords.s_C_LAST,
    SIZ(ordP->in_ords.s_C_LAST),SQLT_STR);
OCIDFNRA(octx->curo0,octx->c_rowid_dp,(OCIErr
*)errhp,1,octx->c_rowid_ptr,
    SIZ(OCIRowid*),SQLT_RDD,NULL,octx->c_rowid_len,
    NULL);

    OCIBND(octx->curo1,octx->c_rowid_bp,(OCIErr
*)errhp,"cust_rowid",&octx->c_rowid_cust,
    sizeof(octx->c_rowid_ptr[0]),SQLT_RDD);
    OCIDDEF(octx->curo1,octx->c_id_dp,(OCIErr
*)errhp,1,ADR(ordP->in_ords.s_C_ID),SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER
    OCIDDEF(octx->curo1,octx->c_balance_dp[0],(OCIErr
*)errhp,2,ADR(ordP->out_ords.s_C_BALANCE),
    SIZ(double),SQLT_BDOUBLE);
#else
    OCIDDEF(octx->curo1,octx->c_balance_dp[0],(OCIErr
*)errhp,2,ADR(ordP->out_ords.s_C_BALANCE),
    SIZ(double),SQLT_FLT);
#endif
    OCIDDEF(octx->curo1,octx->c_first_dp[0],(OCIErr *)errhp,3,ordP-
>out_ords.s_C_FIRST,SIZ(ordP->out_ords.s_C_FIRST)-1,
    SQLT_CHR);
    OCIDDEF(octx->curo1,octx->c_middle_dp[0],(OCIErr
*)errhp,4,ordP->out_ords.s_C_MIDDLE,
    SIZ(ordP->out_ords.s_C_MIDDLE)-1,SQLT_AFC);
    OCIDDEF(octx->curo1,octx->c_last_dp[0],(OCIErr *)errhp,5,ordP-
>out_ords.s_C_LAST,SIZ(ordP->out_ords.s_C_LAST)-1,
    SQLT_CHR);
    OCIDDEF(octx->curo1,octx->o_id_dp[0],(OCIErr
*)errhp,6,ADR(ordP->out_ords.s_O_ID),SIZ(int),SQLT_INT);
    OCIDDEF(octx->curo1,octx->o_entry_d_dp[0],(OCIErr *)errhp,7,
    &ordP-
>out_ords.s_OL_DELIVERY_D_base,SIZ(OCIDate),SQLT_ODT);
    OCIDDEF(octx->curo1,octx->o_cr_id_dp[0],(OCIErr
*)errhp,8,ADR(ordP->out_ords.s_O_CARRIER_ID),
    SIZ(int),SQLT_INT);

```

```

    OCIDDEF(octx->curo1,octx->o_ol_cnt_dp[0],(OCIErr
*)errhp,9,ADR(ordP->out_ords.s_ol_cnt),
    SIZ(int),SQLT_INT);
    OCIDDEF(octx->curo1,octx->o_rowid_dp[0],(OCIErr
*)errhp,10,ADR(octx->o_rowid),
    SIZ(OCIRowid*),SQLT_RDD);

/* Bind for third cursor , no-zero customer id */
    OCIBND(octx->curo2,octx->w_id_bp[1],(OCIErr
*)errhp,"w_id",ADR(ordP->in_ords.s_W_ID),
    SIZ(int),SQLT_INT);
    OCIBND(octx->curo2,octx->d_id_bp[1],(OCIErr
*)errhp,"d_id",ADR(ordP->in_ords.s_D_ID),
    SIZ(int),SQLT_INT);
    OCIBND(octx->curo2,octx->c_id_bp,(OCIErr
*)errhp,"c_id",ADR(ordP->in_ords.s_C_ID),
    SIZ(int),SQLT_INT);
#ifdef USE_IEEE_NUMBER
    OCIDDEF(octx->curo2,octx->c_balance_dp[1],(OCIErr
*)errhp,1,ADR(ordP->out_ords.s_C_BALANCE),
    SIZ(double),SQLT_BDOUBLE);
#else
    OCIDDEF(octx->curo2,octx->c_balance_dp[1],(OCIErr
*)errhp,1,ADR(ordP->out_ords.s_C_BALANCE),
    SIZ(double),SQLT_FLT);
#endif
    OCIDDEF(octx->curo2,octx->c_first_dp[1],(OCIErr *)errhp,2,ordP-
>out_ords.s_C_FIRST,SIZ(ordP->out_ords.s_C_FIRST)-1,
    SQLT_CHR);
    OCIDDEF(octx->curo2,octx->c_middle_dp[1],(OCIErr
*)errhp,3,ordP->out_ords.s_C_MIDDLE,
    SIZ(ordP->out_ords.s_C_MIDDLE)-1,SQLT_AFC);
    OCIDDEF(octx->curo2,octx->c_last_dp[1],(OCIErr *)errhp,4,ordP-
>out_ords.s_C_LAST,SIZ(ordP->out_ords.s_C_LAST)-1,
    SQLT_CHR);
    OCIDDEF(octx->curo2,octx->o_id_dp[1],(OCIErr
*)errhp,5,ADR(ordP->out_ords.s_O_ID),SIZ(int),SQLT_INT);
    OCIDDEF(octx->curo2,octx->o_entry_d_dp[1],(OCIErr *)errhp,6,
    &ordP->out_ords.s_OL_DELIVERY_D_base,
    SIZ(OCIDate),SQLT_ODT);
    OCIDDEF(octx->curo2,octx->o_cr_id_dp[1],(OCIErr
*)errhp,7,ADR(ordP->out_ords.s_O_CARRIER_ID),
    SIZ(int),SQLT_INT);
    OCIDDEF(octx->curo2,octx->o_ol_cnt_dp[1],(OCIErr
*)errhp,8,ADR(ordP->out_ords.s_ol_cnt),
    SIZ(int),SQLT_INT);
    OCIDDEF(octx->curo2,octx->o_rowid_dp[1],(OCIErr
*)errhp,9,ADR(octx->o_rowid),
    SIZ(OCIRowid*),SQLT_RDD);

/* Bind for last cursor */

/*
    OCIBND(octx->curo3,octx->w_id_bp[2],(OCIErr
*)errhp,"w_id",ADR(ordP->ordin.w_id),SIZ(int),SQLT_INT);
    OCIBND(octx->curo3,octx->d_id_bp[2],(OCIErr
*)errhp,"d_id",ADR(ordP->ordin.d_id),SIZ(int),SQLT_INT);
    OCIBND(octx->curo3,octx->o_id_bp,(OCIErr
*)errhp,"o_id",ADR(ordP->ordout.o_id),SIZ(int),SQLT_INT);
    OCIBND(octx->curo3,octx->c_id_bp,(OCIErr
*)errhp,"c_id",ADR(ordP->ordin.c_id),SIZ(int),SQLT_INT);
*/
    OCIBND(octx->curo3,octx->o_rowid_bp,(OCIErr
*)errhp,"ordr_rowid",
    &octx->o_rowid,SIZ(OCIRowid*),SQLT_RDD);

    OCIDFNRA(octx->curo3,octx->ol_i_id_dp,(OCIErr *)errhp,1,
    ordP->out_ords.s_OL_I_ID,SIZ(int),SQLT_INT,
    NULL,octx->ol_i_id_len,NULL);
    OCIDFNRA(octx->curo3,octx->ol_supply_w_id_dp,(OCIErr
*)errhp,2,ordP->out_ords.s_OL_SUPPLY_W_ID,

```

```

        SIZ(int),SQLT_INT, NULL,
        octx->ol_supply_w_id_len, NULL);
#ifdef USE_IEEE_NUMBER
    OCIDFNRA(octx->curo3, octx->ol_quantity_dp, (OCLError
*)errhp, 3, ordP->out_ords.s_OL_QUANTITY, SIZ(float),
    SQLT_FLT, NULL, octx->ol_quantity_len, NULL);
    OCIDFNRA(octx->curo3, octx->ol_amount_dp, (OCLError
*)errhp, 4, ordP->out_ords.s_OL_AMOUNT, SIZ(float),
    SQLT_FLT, NULL, octx->ol_amount_len, NULL);
#else
    OCIDFNRA(octx->curo3, octx->ol_quantity_dp, (OCLError
*)errhp, 3, ordP->out_ords.s_OL_QUANTITY, SIZ(int),
    SQLT_INT, NULL, octx->ol_quantity_len, NULL);
    OCIDFNRA(octx->curo3, octx->ol_amount_dp, (OCLError
*)errhp, 4, ordP->out_ords.s_OL_AMOUNT, SIZ(int),
    SQLT_INT, NULL, octx->ol_amount_len, NULL);
#endif
    OCIDFNRA(octx->curo3, octx->ol_d_base_dp, (OCLError
*)errhp, 5, ordP->out_ords.s_OL_DELIVERY_D_base, SIZ(OCIDate),
    SQLT_ODT, NULL, octx->ol_delivery_d_len, NULL);

    OCIBND(octx->curo4, octx->w_id_bp[3], (OCLError
*)errhp, "w_id", ADR(ordP->in_ords.s_W_ID),
    SIZ(int), SQLT_INT);
    OCIBND(octx->curo4, octx->d_id_bp[3], (OCLError
*)errhp, "d_id", ADR(ordP->in_ords.s_D_ID),
    SIZ(int), SQLT_INT);
    OCIBND(octx->curo4, octx->c_last_bp[1], (OCLError
*)errhp, "c_last", ordP->out_ords.s_C_LAST,
    SIZ(ordP->out_ords.s_C_LAST), SQLT_STR);
    OCIDEF(octx->curo4, octx->c_count_dp, (OCLError
*)errhp, 1, ADR(octx->rcount), SIZ(int),
    SQLT_INT);

    return (0);
}

tkvco (ora_cn_data_t *ora_SlotDataP)
{
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;

    int i;
    int rcount;

    ordctx *octx = (ordctx *)ora_SlotDataP->octx;
    defctx *cbctx = (defctx *)ora_SlotDataP->cbctx;
    ords_wrapper *ordP = &ora_SlotDataP->ordP;
    OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCISmt *curi = ora_SlotDataP->curi;

#ifdef defined(ISO9)
    int secondread = 0;
    char sdate[30];
    ub4 datelen;
    sysdate(sdate);
    printf("Order Status started at: %s\n", sdate);
#endif

    for (i = 0; i < NITEMS; i++) {
        octx->ol_supply_w_id_len[i] = sizeof(int);
        octx->ol_i_id_len[i] = sizeof(int);
#ifdef USE_IEEE_NUMBER
        octx->ol_quantity_len[i] = sizeof(float);
        octx->ol_amount_len[i] = sizeof(float);

```

```

    } else
        octx->ol_quantity_len[i] = sizeof(int);
        octx->ol_amount_len[i] = sizeof(int);
    }
    octx->ol_delivery_d_len[i] = sizeof(OCIDate);
}
octx->ol_supply_w_id_csize = NITEMS;
octx->ol_i_id_csize = NITEMS;
octx->ol_quantity_csize = NITEMS;
octx->ol_amount_csize = NITEMS;
octx->ol_delivery_d_csize = NITEMS;

retry:
    if (ordP->in_ords.bylastname)
    {
        cbctx->reexec = FALSE;
        ordP->out_ords.terror=OCIStmtExecute(tpcsvc, octx-
>curo0, (OCLError *)errhp, 100, 0,
            NULLP(CONST
OCISnapshot), NULLP(OCISnapshot), OCI_DEFAULT);
        /* will get OCI_NO_DATA if <100 found */
        if ((ordP->out_ords.terror != OCI_NO_DATA) && (ordP-
>out_ords.terror != OCI_SUCCESS))
        {
            ordP->out_ords.terror=OCIERROR(errhp, ordP-
>out_ords.terror);
            if ((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP-
>out_ords.terror == RECOVER))
            {
                DISCARD OCITransCommit(tpcsvc, (OCLError
*)errhp, OCI_DEFAULT);
                ordP->out_ords.retry++;
                goto retry;
            } else {
                return -1;
            }
        }
        if (ordP->out_ords.terror == OCI_NO_DATA) /* there are no more
rows */
        {
            /* get rowcount, find middle one */
            DISCARD OCIAttrGet(octx-
>curo0, OCI_HTYPE_STMT, &rcount, NULL,
                OCI_ATTR_ROW_COUNT, (OCLError *)errhp);

            if (rcount < 1)
            {
                oralogprintf(server_logtrans, "ORDERSTATUS:
rcount=%d\n", rcount);
                oralogprintf(server_logtrans, "ORDERSTATUS: c_d_id %d
c_w_id %d c_last %s\n", ordP->in_ords.s_D_ID, ordP-
>in_ords.s_W_ID, ordP->in_ords.s_C_LAST);
                return (-1);
            }
            octx->cust_idx=(rcount)/2 ;
        }
        else
        {
            /* count the number of rows */
            ordP->out_ords.terror=OCIStmtExecute(tpcsvc, octx-
>curo4, (OCLError *)errhp, 1, 0,
                NULLP(CONST
OCISnapshot), NULLP(OCISnapshot), OCI_DEFAULT);
            if ((ordP->out_ords.terror != OCI_NO_DATA) && (ordP-
>out_ords.terror != OCI_SUCCESS))
            {
                ordP->out_ords.terror=OCIERROR(errhp, ordP-
>out_ords.terror);
                if ((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP-
>out_ords.terror == RECOVER))
                {
                    DISCARD OCITransCommit(tpcsvc, (OCLError
*)errhp, OCI_DEFAULT);

```

```

        ordP->out_ords.retry++;
        goto retry;
    } else {
        return -1;
    }
}
if (octx->rcount+1 < 2*10)
    octx->cust_idx=(octx->rcount+1)/2;
else /* */
{
    cbctx->reexec = TRUE;
    cbctx->count = (octx->rcount+1)/2;
    ordP->out_ords.terror=OCIStmtExecute(tpcsvc,octx-
>curo0,(OCIError *)errhp,cbctx->count,
        0,NULLP(CONST OCISnapshot),
        NULLP(OCISnapshot),OCI_DEFAULT);
    /* will get OCI_NO_DATA if <100 found */
    if (cbctx->count > 0)
    {
        oralogprintf(server_logtrans,"ORDERSTATUS: did not get all
rows ");
        return (-1);
    }

    if ((ordP->out_ords.terror != OCI_NO_DATA) && (ordP-
>out_ords.terror != OCI_SUCCESS))
    {
        ordP->out_ords.terror=OCIERROR(errhp, ordP-
>out_ords.terror);
        if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP-
>out_ords.terror == RECOVER))
        {
            DISCARD OCITransCommit(tpcsvc,(OCIError
*)errhp,OCI_DEFAULT);
            ordP->out_ords.retry++;
            goto retry;
        } else {
            return -1;
        }
    }
    octx->cust_idx=0;
}

octx->c_rowid_cust = octx->c_rowid_ptr[octx->cust_idx];
ordP->out_ords.terror=OCIStmtExecute(tpcsvc,octx-
>curo1,(OCIError *)errhp,1,0,
    NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),OCI_DEFAULT);
if (ordP->out_ords.terror != OCI_SUCCESS)
{
    ordP->out_ords.terror=OCIERROR(errhp,ordP-
>out_ords.terror);
    DISCARD OCITransCommit(tpcsvc,(OCIError
*)errhp,OCI_DEFAULT);
    if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP-
>out_ords.terror == RECOVER))
        || (ordP->out_ords.terror == SNAPSHOT_TOO_OLD))
    {
        ordP->out_ords.retry++;
        goto retry;
    } else {
        return -1;
    }
}
} /* lastname */
else
{
    ordP->out_ords.terror=OCIStmtExecute(tpcsvc,octx-
>curo2,(OCIError *)errhp,1,0,
        NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),

```

```

        OCI_DEFAULT);
    if (ordP->out_ords.terror != OCI_SUCCESS)
    {
        ordP->out_ords.terror=OCIERROR(errhp,ordP-
>out_ords.terror);
        DISCARD OCITransCommit(tpcsvc,(OCIError
*)errhp,OCI_DEFAULT);
        if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP-
>out_ords.terror == RECOVER))
            || (ordP->out_ords.terror == SNAPSHOT_TOO_OLD))
        {
            ordP->out_ords.retry++;
            goto retry;
        }
        else
        {
            return -1;
        }
    }
    octx->ol_w_id_len = sizeof(int);
    octx->ol_d_id_len = sizeof(int);
    octx->ol_o_id_len = sizeof(int);

    ordP->out_ords.terror = OCIStmtExecute(tpcsvc,octx-
>curo3,(OCIError *)errhp,ordP->out_ords.s_ol_cnt,0,
        NULLP(CONST
OCISnapshot),NULLP(OCISnapshot),
        OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
    if (ordP->out_ords.terror != OCI_SUCCESS)
    {
        ordP->out_ords.terror=OCIERROR(errhp,ordP-
>out_ords.terror);
        DISCARD OCITransCommit(tpcsvc,(OCIError
*)errhp,OCI_DEFAULT);
        if((ordP->out_ords.terror == NOT_SERIALIZABLE) || (ordP-
>out_ords.terror == RECOVER))
            || (ordP->out_ords.terror == SNAPSHOT_TOO_OLD))
        {
            ordP->out_ords.retry++;
            goto retry;
        }
        else
        {
            return -1;
        }
    }
    /* clean up and convert the delivery dates */
    for (i = 0; i < ordP->out_ords.s_ol_cnt; i++)
    {
        ordP->out_ords.ol_del_len[i]=sizeof(ordP-
>out_ords.s_OL_DELIVERY_D_time[i]);
        DISCARD OCIERROR(errhp,OCIDateToText((OCIError
*)errhp,&ordP->out_ords.s_OL_DELIVERY_D_base[i],
            (const
text*)SHORTDATE,(ub1)strlen(SHORTDATE),(text*)0,0,
            (ub4 *)&ordP->out_ords.ol_del_len[i], (text *)ordP-
>out_ords.s_OL_DELIVERY_D_time[i]));
    }
    ordP->out_ords.terror = 0;
    return (0);
}

void tkvcodone (ora_cn_data_t *ora_SlotDataP)
{
    if (ora_SlotDataP->octx) {
        free (ora_SlotDataP->octx);
        ora_SlotDataP->octx = NULL;
    }
}

```

```

}
}

```

```

/* end of file tkvcord.c */

```

plpay.cpp

```

#include "../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"

```

```

extern char OracleHome[256];
extern int sqlfile(FILE *server_logtrans, char *fnam, text *linebuf);

```

```

extern void oralogprintf(FILE *, char *format, ...);

```

```

#define SQLTXT_INIT "BEGIN inittpcc.init_pay; END;"

```

```

struct payctx {
    OCIStmt *curpi;
    OCIStmt *curp0;
    OCIStmt *curp1;
    OCIBind *w_id_bp[2];
    ub2 w_id_len;

    OCIBind *d_id_bp[2];
    ub2 d_id_len;

    OCIBind *c_w_id_bp[2];
    ub2 c_w_id_len;

    OCIBind *c_d_id_bp[2];
    ub2 c_d_id_len;

    OCIBind *c_id_bp[2];
    ub2 c_id_len;

    OCIBind *h_amount_bp[2];
    ub2 h_amount_len;

    OCIBind *c_last_bp[2];
    ub2 c_last_len;

    OCIBind *w_street_1_bp[2];
    ub2 w_street_1_len;

    OCIBind *w_street_2_bp[2];
    ub2 w_street_2_len;

    OCIBind *w_city_bp[2];
    ub2 w_city_len;

    OCIBind *w_state_bp[2];
    ub2 w_state_len;

    OCIBind *w_zip_bp[2];
    ub2 w_zip_len;

    OCIBind *d_street_1_bp[2];
    ub2 d_street_1_len;

    OCIBind *d_street_2_bp[2];
    ub2 d_street_2_len;

    OCIBind *d_city_bp[2];
    ub2 d_city_len;

    OCIBind *d_state_bp[2];
    ub2 d_state_len;

```

```

OCIBind *d_zip_bp[2];
ub2 d_zip_len;

```

```

OCIBind *c_first_bp[2];
ub2 c_first_len;

```

```

OCIBind *c_middle_bp[2];
ub2 c_middle_len;

```

```

OCIBind *c_street_1_bp[2];
ub2 c_street_1_len;

```

```

OCIBind *c_street_2_bp[2];
ub2 c_street_2_len;

```

```

OCIBind *c_city_bp[2];
ub2 c_city_len;

```

```

OCIBind *c_state_bp[2];
ub2 c_state_len;

```

```

OCIBind *c_zip_bp[2];
ub2 c_zip_len;

```

```

OCIBind *c_phone_bp[2];
ub2 c_phone_len;

```

```

OCIBind *c_since_bp[2];
ub2 c_since_len;

```

```

OCIBind *c_credit_bp[2];
ub2 c_credit_len;

```

```

OCIBind *c_credit_lim_bp[2];
ub2 c_credit_lim_len;

```

```

OCIBind *c_discount_bp[2];
ub2 c_discount_len;

```

```

OCIBind *c_balance_bp[2];
ub2 c_balance_len;

```

```

OCIBind *c_data_bp[2];
ub2 c_data_len;

```

```

OCIBind *h_date_bp[2];
ub2 h_date_len;

```

```

OCIBind *retries_bp[2];
ub2 retries_len;

```

```

OCIBind *cr_date_bp[2];
ub2 cr_date_len;

```

```

OCIBind *byln_bp[2];
ub2 byln_len;

```

```

};

```

```

typedef struct payctx payctx;

```

```

int tkvcpinit (ora_cn_data_t *ora_SlotDataP)
{
    char sqlfilename[256];
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;
    /* */
    payctx *pctx;
    paym_wrapper *payP;
    OCIEEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;

```

```

OCIStmt *curi = ora_SlotDataP->curi;

oralogprintf(ora_SlotDataP->server_logtrans,">>tkvcpinit\n");
text stmbuf[SQL_BUF_SIZE];

pctx = (payctx *)malloc(sizeof(payctx));
memset(pctx, (char)0, sizeof(payctx));
ora_SlotDataP->pctx = (void *)pctx;

payP = &ora_SlotDataP->payP;
memset(&ora_SlotDataP->payP, (char)0, sizeof(paym_wrapper));

/* cursor for init */
DISCARD OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid
**)(&(pctx->curpi)),
OCI_HTYPE_STMT, 0, (dvoid**)0));

DISCARD OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid
**)(&(pctx->curp0)),
OCI_HTYPE_STMT, 0, (dvoid**)0));
DISCARD OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid
**)(&(pctx->curp1)),
OCI_HTYPE_STMT, 0, (dvoid**)0));

/* build the init statement and execute it */

sprintf((char*)stmbuf, SQLTXT_INIT);
DISCARD OCIERROR(errhp, OCIStmtPrepare(pctx->curpi,
(OCIError *)errhp, stmbuf,
strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT));
DISCARD OCIERROR(errhp, OCIStmtExecute(tpcenv, pctx-
>curpi, (OCIError *)errhp, 1, 0,
NULLP(CONST
OCISnapshot), NULLP(OCISnapshot), OCI_DEFAULT));

/* customer id != 0, go by cust id */
sprintf(sqlfilename, "%s/tpcc-
kit/benchrun/blocks/paynz.sql", OracleHome);
sqlfile(ora_SlotDataP->server_logtrans, sqlfilename, stmbuf);
DISCARD OCIERROR(errhp, OCIStmtPrepare(pctx->curp0,
(OCIError *)errhp, stmbuf,
strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT));

/* customer id == 0, go by last name */

sprintf(sqlfilename, "%s/tpcc-
kit/benchrun/blocks/payz.sql", OracleHome);
sqlfile(ora_SlotDataP->server_logtrans, sqlfilename, stmbuf);
DISCARD OCIERROR(errhp, OCIStmtPrepare(pctx->curp1,
(OCIError *)errhp, stmbuf,
strlen((char *)stmbuf), OCI_NTV_SYNTAX,
OCI_DEFAULT));

pctx->w_id_len = SIZ(payP->in_paym.s_W_ID);
pctx->d_id_len = SIZ(payP->in_paym.s_D_ID);
pctx->c_w_id_len = SIZ(payP->in_paym.s_C_W_ID);
pctx->c_d_id_len = SIZ(payP->in_paym.s_C_D_ID);
pctx->c_id_len = 0;
pctx->h_amount_len = SIZ(int);
pctx->c_last_len = 0;
pctx->w_street_1_len = 0;
pctx->w_street_2_len = 0;
pctx->w_city_len = 0;
pctx->w_state_len = 0;
pctx->w_zip_len = 0;
pctx->d_street_1_len = 0;
pctx->d_street_2_len = 0;
pctx->d_city_len = 0;
pctx->d_state_len = 0;
pctx->d_zip_len = 0;

```

```

pctx->c_first_len = 0;
pctx->c_middle_len = 0;
pctx->c_street_1_len = 0;
pctx->c_street_2_len = 0;
pctx->c_city_len = 0;
pctx->c_state_len = 0;
pctx->c_zip_len = 0;
pctx->c_phone_len = 0;
pctx->c_since_len = 7;
pctx->c_credit_len = 0;
pctx->c_credit_lim_len = 0;
pctx->c_discount_len = 0;
pctx->c_balance_len = sizeof(double);
pctx->c_data_len = 0;
pctx->h_date_len = 0;
pctx->retries_len = SIZ(payP->out_paym.retry);
pctx->cr_date_len = 7;

```

/* bind variables */

```

OCIBNDPL(pctx->curp0, pctx->w_id_bp[0], (OCIError
*)errhp, "w_id", ADR(payP->in_paym.s_W_ID), SIZ(int),
SQLT_INT, NULL);

OCIBNDPL(pctx->curp0, pctx->d_id_bp[0], (OCIError
*)errhp, "d_id", ADR(payP->in_paym.s_D_ID), SIZ(int),
SQLT_INT, NULL);
OCIBND(pctx->curp0, pctx->c_w_id_bp[0], (OCIError
*)errhp, "c_w_id", ADR(payP->in_paym.s_C_W_ID), SIZ(int),
SQLT_INT);
OCIBND(pctx->curp0, pctx->c_d_id_bp[0], (OCIError
*)errhp, "c_d_id", ADR(payP->in_paym.s_C_D_ID), SIZ(int),
SQLT_INT);
OCIBND(pctx->curp0, pctx->c_id_bp[0], (OCIError
*)errhp, "c_id", ADR(payP->in_paym.s_C_ID), SIZ(int),
SQLT_INT);

OCIBNDPL(pctx->curp0, pctx->h_amount_bp[0], (OCIError
*)errhp, "h_amount", ADR(payP->in_paym.s_H_AMOUNT),
SIZ(payP->in_paym.s_H_AMOUNT), SQLT_INT, &pctx-
>h_amount_len);

OCIBNDPL(pctx->curp0, pctx->c_last_bp[0], (OCIError
*)errhp, "c_last", payP->out_paym.s_C_LAST, SIZ(payP-
>out_paym.s_C_LAST),
SQLT_STR, &pctx->c_last_len);

OCIBNDPL(pctx->curp0, pctx->w_street_1_bp[0], (OCIError
*)errhp, "w_street_1", payP->out_paym.s_W_STREET_1,
SIZ(payP->out_paym.s_W_STREET_1), SQLT_STR, &pctx-
>w_street_1_len);
OCIBNDPL(pctx->curp0, pctx->w_street_2_bp[0], (OCIError
*)errhp, "w_street_2", payP->out_paym.s_W_STREET_2,
SIZ(payP->out_paym.s_W_STREET_2), SQLT_STR, &pctx-
>w_street_2_len);
OCIBNDPL(pctx->curp0, pctx->w_city_bp[0], (OCIError
*)errhp, "w_city", payP->out_paym.s_W_CITY, SIZ(payP-
>out_paym.s_W_CITY),
SQLT_STR, &pctx->w_city_len);
OCIBNDPL(pctx->curp0, pctx->w_state_bp[0], (OCIError
*)errhp, "w_state", payP->out_paym.s_W_STATE,
SIZ(payP->out_paym.s_W_STATE), SQLT_STR, &pctx-
>w_state_len);
OCIBNDPL(pctx->curp0, pctx->w_zip_bp[0], (OCIError
*)errhp, "w_zip", payP->out_paym.s_W_ZIP, SIZ(payP-
>out_paym.s_W_ZIP),
SQLT_STR, &pctx->w_zip_len);

```

```

OCIBNDPL(pctx->curp0, pctx->d_street_1_bp[0], (OCLError
*)errhp, "d_street_1", payP->out_paym.s_D_STREET_1,
SIZ(payP->out_paym.s_D_STREET_1), SQLT_STR, &pctx-
>d_street_1_len);
OCIBNDPL(pctx->curp0, pctx->d_street_2_bp[0], (OCLError
*)errhp, "d_street_2", payP->out_paym.s_D_STREET_2,
SIZ(payP->out_paym.s_D_STREET_2), SQLT_STR, &pctx-
>d_street_2_len);
OCIBNDPL(pctx->curp0, pctx->d_city_bp[0], (OCLError
*)errhp, "d_city", payP->out_paym.s_D_CITY, SIZ(payP-
>out_paym.s_D_CITY),
SQLT_STR, &pctx->d_city_len);
OCIBNDPL(pctx->curp0, pctx->d_state_bp[0], (OCLError
*)errhp, "d_state", payP->out_paym.s_D_STATE,
SIZ(payP->out_paym.s_D_STATE), SQLT_STR, &pctx-
>d_state_len);
OCIBNDPL(pctx->curp0, pctx->d_zip_bp[0], (OCLError
*)errhp, "d_zip", payP->out_paym.s_D_ZIP, SIZ(payP-
>out_paym.s_D_ZIP),
SQLT_STR, &pctx->d_zip_len);
OCIBNDPL(pctx->curp0, pctx->c_first_bp[0], (OCLError
*)errhp, "c_first", payP->out_paym.s_C_FIRST,
SIZ(payP->out_paym.s_C_FIRST), SQLT_STR, &pctx-
>c_first_len);
OCIBNDPL(pctx->curp0, pctx->c_middle_bp[0], (OCLError
*)errhp, "c_middle", payP->out_paym.s_C_MIDDLE, 2,
SQLT_AFC, &pctx->c_middle_len);
OCIBNDPL(pctx->curp0, pctx->c_street_1_bp[0], (OCLError
*)errhp, "c_street_1", payP->out_paym.s_C_STREET_1,
SIZ(payP->out_paym.s_C_STREET_1), SQLT_STR, &pctx-
>c_street_1_len);
OCIBNDPL(pctx->curp0, pctx->c_street_2_bp[0], (OCLError
*)errhp, "c_street_2", payP->out_paym.s_C_STREET_2,
SIZ(payP->out_paym.s_C_STREET_2), SQLT_STR, &pctx-
>c_street_2_len);
OCIBNDPL(pctx->curp0, pctx->c_city_bp[0], (OCLError
*)errhp, "c_city", payP->out_paym.s_C_CITY, SIZ(payP-
>out_paym.s_C_CITY),
SQLT_STR, &pctx->c_city_len);
OCIBNDPL(pctx->curp0, pctx->c_state_bp[0], (OCLError
*)errhp, "c_state", payP->out_paym.s_C_STATE,
SIZ(payP->out_paym.s_C_STATE), SQLT_STR, &pctx-
>c_state_len);
OCIBNDPL(pctx->curp0, pctx->c_zip_bp[0], (OCLError
*)errhp, "c_zip", payP->out_paym.s_C_ZIP, SIZ(payP-
>out_paym.s_C_ZIP),
SQLT_STR, &pctx->c_zip_len);
OCIBNDPL(pctx->curp0, pctx->c_phone_bp[0], (OCLError
*)errhp, "c_phone", payP->out_paym.s_C_PHONE,
SIZ(payP->out_paym.s_C_PHONE), SQLT_STR, &pctx-
>c_phone_len);
OCIBNDPL(pctx->curp0, pctx->c_since_bp[0], (OCLError
*)errhp, "c_since", &payP->out_paym.c_since,
SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
OCIBNDPL(pctx->curp0, pctx->c_credit_bp[0], (OCLError
*)errhp, "c_credit", payP->out_paym.s_C_CREDIT,
SIZ(payP->out_paym.s_C_CREDIT), SQLT_CHR, &pctx-
>c_credit_len);

OCIBNDPL(pctx->curp0, pctx->c_credit_lim_bp[0], (OCLError
*)errhp, "c_credit_lim",
ADR(payP->out_paym.s_C_CREDIT_LIM), SIZ(double),
SQLT_FLT, &pctx->c_credit_lim_len);

OCIBNDPL(pctx->curp0, pctx->c_discount_bp[0], (OCLError
*)errhp, "c_discount",
ADR(payP->out_paym.s_C_DISCOUNT), SIZ(payP-
>out_paym.s_C_DISCOUNT), SQLT_FLT, &pctx->c_discount_len);
OCIBNDPL(pctx->curp0, pctx->c_balance_bp[0], (OCLError
*)errhp, "c_balance",
ADR(payP->out_paym.s_C_BALANCE),
SIZ(double), SQLT_FLT, &pctx->c_balance_len);

```

```

OCIBNDPL(pctx->curp0, pctx->c_data_bp[0], (OCLError
*)errhp, "c_data", payP->out_paym.s_C_DATA, SIZ(payP-
>out_paym.s_C_DATA),
SQLT_STR, &pctx->c_data_len);

```

```

OCIBNDPL(pctx->curp0, pctx->retries_bp[0], (OCLError
*)errhp, "retry", ADR(payP->out_paym.retry),
SIZ(int), SQLT_INT, &pctx->retries_len);

```

```

OCIBNDPL(pctx->curp0, pctx->cr_date_bp[0], (OCLError
*)errhp, "cr_date", ADR(payP->in_paym.cr_date),
SIZ(OCIDate), SQLT_ODT, &pctx->cr_date_len);

```

/* ---- Binds for the second cursor */

```

OCIBNDPL(pctx->curp1, pctx->w_id_bp[1], (OCLError
*)errhp, "w_id", ADR(payP->in_paym.s_W_ID), SIZ(int),
SQLT_INT, &pctx->w_id_len);
OCIBNDPL(pctx->curp1, pctx->d_id_bp[1], (OCLError
*)errhp, "d_id", ADR(payP->in_paym.s_D_ID), SIZ(int),
SQLT_INT, &pctx->d_id_len);
OCIBNDPL(pctx->curp1, pctx->c_w_id_bp[1], (OCLError
*)errhp, "c_w_id", ADR(payP->in_paym.s_C_W_ID), SIZ(int),
SQLT_INT);
OCIBNDPL(pctx->curp1, pctx->c_d_id_bp[1], (OCLError
*)errhp, "c_d_id", ADR(payP->in_paym.s_C_D_ID), SIZ(int),
SQLT_INT);
OCIBNDPL(pctx->curp1, pctx->c_id_bp[1], (OCLError
*)errhp, "c_id", ADR(payP->out_paym.s_C_ID), SIZ(int),
SQLT_INT, &pctx->c_id_len);
#ifdef USE_IEEE_NUMBER
OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1], (OCLError
*)errhp, "h_amount", ADR(payP->in_paym.s_H_AMOUNT),
SIZ(float), SQLT_FLT, &pctx->h_amount_len);
#else
OCIBNDPL(pctx->curp1, pctx->h_amount_bp[1], (OCLError
*)errhp, "h_amount", ADR(payP->in_paym.s_H_AMOUNT),
SIZ(int), SQLT_INT, &pctx->h_amount_len);
#endif
OCIBNDPL(pctx->curp1, pctx->c_last_bp[1], (OCLError
*)errhp, "c_last", payP->in_paym.s_C_LAST, SIZ(payP-
>in_paym.s_C_LAST),
SQLT_STR);
OCIBNDPL(pctx->curp1, pctx->w_street_1_bp[1], (OCLError
*)errhp, "w_street_1", payP->out_paym.s_W_STREET_1,
SIZ(payP->out_paym.s_W_STREET_1), SQLT_STR, &pctx-
>w_street_1_len);
OCIBNDPL(pctx->curp1, pctx->w_street_2_bp[1], (OCLError
*)errhp, "w_street_2", payP->out_paym.s_W_STREET_2,
SIZ(payP->out_paym.s_W_STREET_2), SQLT_STR, &pctx-
>w_street_2_len);
OCIBNDPL(pctx->curp1, pctx->w_city_bp[1], (OCLError
*)errhp, "w_city", payP->out_paym.s_W_CITY, SIZ(payP-
>out_paym.s_W_CITY),
SQLT_STR, &pctx->w_city_len);
OCIBNDPL(pctx->curp1, pctx->w_state_bp[1], (OCLError
*)errhp, "w_state", payP->out_paym.s_W_STATE,
SIZ(payP->out_paym.s_W_STATE), SQLT_STR, &pctx-
>w_state_len);
OCIBNDPL(pctx->curp1, pctx->w_zip_bp[1], (OCLError
*)errhp, "w_zip", payP->out_paym.s_W_ZIP, SIZ(payP-
>out_paym.s_W_ZIP),
SQLT_STR, &pctx->w_zip_len);
OCIBNDPL(pctx->curp1, pctx->d_street_1_bp[1], (OCLError
*)errhp, "d_street_1", payP->out_paym.s_D_STREET_1,
SIZ(payP->out_paym.s_D_STREET_1), SQLT_STR, &pctx-
>d_street_1_len);
OCIBNDPL(pctx->curp1, pctx->d_street_2_bp[1], (OCLError
*)errhp, "d_street_2", payP->out_paym.s_D_STREET_2,

```



```

        SIZ(payP->out_paym.s_D_STREET_2),SQLT_STR, &pctx-
>d_street_2_len);
        OCIBNDPL(pctx->curp1, pctx->d_city_bp[1], (OCLError
*)errhp,":d_city",payP->out_paym.s_D_CITY,SIZ(payP-
>out_paym.s_D_CITY),
        SQLT_STR, &pctx->d_city_len);
        OCIBNDPL(pctx->curp1, pctx->d_state_bp[1], (OCLError
*)errhp,":d_state",payP->out_paym.s_D_STATE,
        SIZ(payP->out_paym.s_D_STATE), SQLT_STR, &pctx-
>d_state_len);
        OCIBNDPL(pctx->curp1, pctx->d_zip_bp[1], (OCLError
*)errhp,":d_zip",payP->out_paym.s_D_ZIP,SIZ(payP-
>out_paym.s_D_ZIP),
        SQLT_STR, &pctx->d_zip_len);
        OCIBNDPL(pctx->curp1, pctx->c_first_bp[1], (OCLError
*)errhp,":c_first",payP->out_paym.s_C_FIRST,
        SIZ(payP->out_paym.s_C_FIRST), SQLT_STR, &pctx-
>c_first_len);
        OCIBNDPL(pctx->curp1, pctx->c_middle_bp[1], (OCLError
*)errhp,":c_middle",payP->out_paym.s_C_MIDDLE,2,
        SQLT_AFC, &pctx->c_middle_len);

        OCIBNDPL(pctx->curp1, pctx->c_street_1_bp[1], (OCLError
*)errhp,":c_street_1",payP->out_paym.s_C_STREET_1,
        SIZ(payP->out_paym.s_C_STREET_1),SQLT_STR, &pctx-
>c_street_1_len);
        OCIBNDPL(pctx->curp1, pctx->c_street_2_bp[1], (OCLError
*)errhp,":c_street_2",payP->out_paym.s_C_STREET_2,
        SIZ(payP->out_paym.s_C_STREET_2),SQLT_STR, &pctx-
>c_street_2_len);
        OCIBNDPL(pctx->curp1, pctx->c_city_bp[1], (OCLError
*)errhp,":c_city",payP->out_paym.s_C_CITY,
        SIZ(payP->out_paym.s_C_CITY),SQLT_STR, &pctx-
>c_city_len);
        OCIBNDPL(pctx->curp1, pctx->c_state_bp[1], (OCLError
*)errhp,":c_state",payP->out_paym.s_C_STATE,
        SIZ(payP->out_paym.s_C_STATE), SQLT_STR, &pctx-
>c_state_len);
        OCIBNDPL(pctx->curp1, pctx->c_zip_bp[1], (OCLError
*)errhp,":c_zip",payP->out_paym.s_C_ZIP,SIZ(payP-
>out_paym.s_C_ZIP),
        SQLT_STR, &pctx->c_zip_len);
        OCIBNDPL(pctx->curp1, pctx->c_phone_bp[1], (OCLError
*)errhp,":c_phone",payP->out_paym.s_C_PHONE,
        SIZ(payP->out_paym.s_C_PHONE), SQLT_STR, &pctx-
>c_phone_len);
        OCIBNDPL(pctx->curp1, pctx->c_since_bp[1], (OCLError
*)errhp,":c_since",&payP->out_paym.c_since,
        SIZ(OCIDate), SQLT_ODT, &pctx->c_since_len);
        OCIBNDPL(pctx->curp1, pctx->c_credit_bp[1], (OCLError
*)errhp,":c_credit",payP->out_paym.s_C_CREDIT,
        SIZ(payP->out_paym.s_C_CREDIT),SQLT_CHR, &pctx-
>c_credit_len);
        OCIBNDPL(pctx->curp1, pctx->c_credit_lim_bp[1], (OCLError
*)errhp,":c_credit_lim",
        ADR(payP->out_paym.s_C_CREDIT_LIM),SIZ(double),
SQLT_FLT, &pctx->c_credit_lim_len);
        OCIBNDPL(pctx->curp1, pctx->c_discount_bp[1], (OCLError
*)errhp,":c_discount",
        ADR(payP->out_paym.s_C_DISCOUNT),SIZ(float),
SQLT_FLT, &pctx->c_discount_len);

        OCIBNDPL(pctx->curp1, pctx->c_balance_bp[1], (OCLError
*)errhp,":c_balance",
        ADR(payP->out_paym.s_C_BALANCE),
SIZ(double),SQLT_FLT, &pctx->c_balance_len);
        OCIBNDPL(pctx->curp1, pctx->c_data_bp[1], (OCLError
*)errhp,":c_data",payP->out_paym.s_C_DATA,SIZ(payP-
>out_paym.s_C_DATA),
        SQLT_STR, &pctx->c_data_len);
        OCIBNDPL(pctx->curp1, pctx->retries_bp[1], (OCLError
*)errhp,":retry",ADR(payP->out_paym.retry),

```

```

        SIZ(int), SQLT_INT, &pctx->retries_len);
        OCIBNDPL(pctx->curp1, pctx->cr_date_bp[1], (OCLError
*)errhp,":cr_date",ADR(payP->in_paym.cr_date),
        SIZ(OCIDate),SQLT_ODT, &pctx->cr_date_len);
        oralogprintf(ora_SlotDataP->server_logtrans,"<<tkvcpinit\n");
        return (0);
}

```

```

tkvcv (ora_cn_data_t *ora_SlotDataP)

```

```

{
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;

```

```

/* */

```

```

payctx *pctx = (payctx *)ora_SlotDataP->pctx;
paym_wrapper *payP = &ora_SlotDataP->payP;
OCISvcCtx *tpcenv = ora_SlotDataP->tpcenv;
OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
dvoid *errhp = ora_SlotDataP->errhp;
OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
OCISession *tpcusr = ora_SlotDataP->tpcusr;
OCISmt *curi = ora_SlotDataP->curi;

```

```

retry:

```

```

pctx->w_id_len = SIZ(payP->in_paym.s_W_ID);
pctx->d_id_len = SIZ(payP->in_paym.s_D_ID);
pctx->c_w_id_len = 0;
pctx->c_d_id_len = 0;
pctx->c_id_len = 0;
pctx->h_amount_len = SIZ(payP->in_paym.s_H_AMOUNT);
pctx->c_last_len = SIZ(payP->in_paym.s_C_LAST);
pctx->w_street_1_len = 0;
pctx->w_street_2_len = 0;
pctx->w_city_len = 0;
pctx->w_state_len = 0;
pctx->w_zip_len = 0;
pctx->d_street_1_len = 0;
pctx->d_street_2_len = 0;
pctx->d_city_len = 0;
pctx->d_state_len = 0;
pctx->d_zip_len = 0;
pctx->c_first_len = 0;
pctx->c_middle_len = 0;
pctx->c_street_1_len = 0;
pctx->c_street_2_len = 0;
pctx->c_city_len = 0;
pctx->c_state_len = 0;
pctx->c_zip_len = 0;
pctx->c_phone_len = 0;
pctx->c_since_len = 7;
pctx->c_credit_len = 0;
pctx->c_credit_lim_len = 0;
pctx->c_discount_len = 0;
pctx->c_balance_len = sizeof(double);
pctx->c_data_len = 0;
pctx->h_date_len = 0;
pctx->retries_len = SIZ(payP->out_paym.retry);
pctx->cr_date_len = 7;

```

```

if(payP->in_paym.bylastname) {
    payP->out_paym.terror=OCISmtExecute(tpcsvc,pctx-
>curp1,(OCLError *)errhp,1,0,
        NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
        OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
} else {
    payP->out_paym.terror=OCISmtExecute(tpcsvc,pctx-
>curp0,(OCLError *)errhp,1,0,
        NULLP(CONST OCISnapshot),NULLP(OCISnapshot),
        OCI_DEFAULT|OCI_COMMIT_ON_SUCCESS);
    payP->out_paym.s_C_ID = payP->in_paym.s_C_ID;
}

```

```

}

if(payP->out_paym.terror != OCI_SUCCESS) {
    oralogprintf(ora_SlotDataP->server_logtrans,"payP-
>out_paym.terror = %d\n",payP->out_paym.terror);
    OCITransRollback(tpcsvc,(OCIError *)errhp,OCI_DEFAULT);
    payP->out_paym.terror = OCIERROR(errhp,payP-
>out_paym.terror);
    oralogprintf(ora_SlotDataP->server_logtrans,"payP-
>out_paym.terror = %d\n",payP->out_paym.terror);
    if(payP->out_paym.terror == NOT_SERIALIZABLE) {
        payP->out_paym.retry++;
        goto retry;
    } else if (payP->out_paym.terror == RECOVERR) {
        payP->out_paym.retry++;
        goto retry;
    } else if (payP->out_paym.terror == SNAPSHOT_TOO_OLD) {
        payP->out_paym.retry++;
        goto retry;
    } else {
        return -1;
    }
}
return 0;
}

```

```
void tkvcpdone (ora_cn_data_t *ora_SlotDataP)
```

```

{
    if(ora_SlotDataP->pctx) {
        free(ora_SlotDataP->pctx);
        ora_SlotDataP->pctx = NULL;
    }
}

```

plsto.cpp

```

#include "../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"

```

```
extern void oralogprintf(FILE *,char *format, ...);
```

```

#ifdef PLSQLSTO
#define SQLTXT "BEGIN stocklevel.getstocklevel
(:w_id, :d_id, :threshold, \
:low_stock); END;"
#else
#define SQLTXT "SELECT /*+ nocache (stok) */ count (DISTINCT
s_i_id) \
FROM ordl, stok, dist \
WHERE d_id = :d_id AND d_w_id = :w_id AND \
d_id = ol_d_id AND d_w_id = ol_w_id AND \
ol_i_id = s_i_id AND ol_w_id = s_w_id AND \
s_quantity < :threshold AND \
ol_o_id BETWEEN (d_next_o_id - 20) AND (d_next_o_id
- 1) \
order by ol_o_id desc"
#endif

```

```

struct stoctx {
    OCIStmt *curs;
    OCIBind *w_id_bp;
    OCIBind *d_id_bp;
    OCIBind *threshold_bp;
#ifdef PLSQLSTO
    OCIBind *low_stock_bp;
#else
    OCIDefine *low_stock_bp;

```

```

#endif
int norow;
};

```

```
typedef struct stoctx stoctx;
```

```
tkvcsinit (ora_cn_data_t *ora_SlotDataP)
```

```

{
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;

    /* */
    stoctx *sctx;
    stok_wrapper *stoP;
    OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCISmt *curi = ora_SlotDataP->curi;

    text stmbuf[SQL_BUF_SIZE];
    sctx = (stoctx *)malloc(sizeof(stoctx));
    memset(sctx,(char)0,sizeof(stoctx));
    ora_SlotDataP->sctx = (void *)sctx;

    memset(&ora_SlotDataP->stoP,(char)0,sizeof(stok_wrapper));
    stoP = &ora_SlotDataP->stoP;

    sctx->norow=0;

    OCIERROR((OCIError *)errhp,
    OCIHandleAlloc(tpcenv,(dvoid**)&sctx-
>curs,OCI_HTYPE_STMT,0,(dvoid**)0));
    sprintf ((char *) stmbuf, SQLTXT);
    OCIERROR((OCIError *)errhp,OCISmtPrepare(sctx-
>curs,(OCIError *)errhp,stmbuf,strlen((char *)stmbuf),
    OCI_NTV_SYNTAX,OCI_DEFAULT));
#ifdef PLSQLSTO
    OCIERROR((OCIError *)errhp,
    OCIAttrSet(sctx->curs,OCI_HTYPE_STMT,(dvoid*)&sctx-
>norow,0,
    OCI_ATTR_PREFETCH_ROWS,(OCIError *)errhp));
#endif

    /* bind variables */

    OCIBND(sctx->curs,sctx->w_id_bp,(OCIError *)errhp, ":w_id",
    ADR(stoP->in_stok.s_W_ID),sizeof(int),
    SQLT_INT);
    OCIBND(sctx->curs,sctx->d_id_bp,(OCIError *)errhp, ":d_id",
    ADR(stoP->in_stok.s_D_ID),sizeof(int),
    SQLT_INT);
#ifdef USE_IEEE_NUMBER
    OCIBND(sctx->curs,sctx->threshold_bp,(OCIError *)errhp,
    ":threshold", ADR(stoP->in_stok.s_threshold),
    sizeof(float),SQLT_FLT);
#else
    OCIBND(sctx->curs,sctx->threshold_bp,(OCIError *)errhp,
    ":threshold", ADR(stoP->in_stok.s_threshold),
    sizeof(int),SQLT_INT);
#endif
#ifdef PLSQLSTO
    OCIBND(sctx->curs,sctx->low_stock_bp,(OCIError
*)errhp,":low_stock", ADR(stoP->out_stok.s_low_stock),
    sizeof(int), SQLT_INT);
#else
    OCIDefine(sctx->curs,sctx->low_stock_bp,(OCIError *)errhp, 1,
    ADR(stoP->out_stok.s_low_stock),
    sizeof(int), SQLT_INT);
#endif
}

```

```

return (0);
}

tkvcs (ora_cn_data_t *ora_SlotDataP)
{
    FILE *server_logtrans = ora_SlotDataP->server_logtrans;
    stoctx *sctx = (stoctx *)ora_SlotDataP->sctx;
    stok_wrapper *stoP = &ora_SlotDataP->stoP;
    OCIEnv *tpcenv = ora_SlotDataP->tpcenv;
    OCIServer *tpcsrv = ora_SlotDataP->tpcsrv;
    dvoid *errhp = ora_SlotDataP->errhp;
    OCISvcCtx *tpcsvc = ora_SlotDataP->tpcsvc;
    OCISession *tpcusr = ora_SlotDataP->tpcusr;
    OCISmt *curi = ora_SlotDataP->curi;

    retry:
        stoP->out_stok.terror= OCISmtExecute(tpcsvc,sctx-
>curs,(OCIErr *errhp,1,0,0,0,
        OCI_COMMIT_ON_SUCCESS | OCI_DEFAULT);
        if (stoP->out_stok.terror != OCI_SUCCESS)
        {
            stoP->out_stok.terror=OCIERROR((OCIErr *)errhp,stoP-
>out_stok.terror);
            OCITransRollback(tpcsvc,(OCIErr *)errhp,OCI_DEFAULT);
            if((stoP->out_stok.terror == NOT_SERIALIZABLE) || (stoP-
>out_stok.terror == RECOVER))
                || (stoP->out_stok.terror == SNAPSHOT_TOO_OLD))
            {
                stoP->out_stok.retry++;
                goto retry;
            } else {
                return -1;
            }
        }

    return (0);
}

void tkvcsdone (ora_cn_data_t *ora_SlotDataP)
{
    /* */
    stoctx *sctx = (stoctx *)ora_SlotDataP->sctx;
    if (sctx) {
        free(sctx);
        ora_SlotDataP->sctx = NULL;
    }
}

```

stdafx.cpp

// stdafx.cpp : source file that includes just the standard includes
// tpccOracleGlue.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

tpccOracleGlue.cpp

// tpccOracleGlue.cpp : Defines the entry point for the DLL
application.
//

```
#include "stdafx.h"
#include "../tpccCom0/tpccCom0.h"
```

```

#include "../tpccCom0/tpccCom0_i.c"
#ifdef PARTITION
#include "../tpccCom1/tpccCom1.h"
#include "../tpccCom1/tpccCom1_i.c"
#include "../tpccCom2/tpccCom2.h"
#include "../tpccCom2/tpccCom2_i.c"
#include "../tpccCom3/tpccCom3.h"
#include "../tpccCom3/tpccCom3_i.c"
#include "../tpccCom4/tpccCom4.h"
#include "../tpccCom4/tpccCom4_i.c"
#include "../tpccCom5/tpccCom5.h"
#include "../tpccCom5/tpccCom5_i.c"
#include "../tpccCom6/tpccCom6.h"
#include "../tpccCom6/tpccCom6_i.c"
#include "../tpccCom7/tpccCom7.h"
#include "../tpccCom7/tpccCom7_i.c"
#include "../tpccCom8/tpccCom8.h"
#include "../tpccCom8/tpccCom8_i.c"
#include "../tpccCom9/tpccCom9.h"
#include "../tpccCom9/tpccCom9_i.c"
#include "../tpccCom10/tpccCom10.h"
#include "../tpccCom10/tpccCom10_i.c"
#include "../tpccCom11/tpccCom11.h"
#include "../tpccCom11/tpccCom11_i.c"
#include "../tpccCom12/tpccCom12.h"
#include "../tpccCom12/tpccCom12_i.c"
#include "../tpccCom13/tpccCom13.h"
#include "../tpccCom13/tpccCom13_i.c"
#include "../tpccCom14/tpccCom14.h"
#include "../tpccCom14/tpccCom14_i.c"
#include "../tpccCom15/tpccCom15.h"
#include "../tpccCom15/tpccCom15_i.c"
#include "../tpccCom16/tpccCom16.h"
#include "../tpccCom16/tpccCom16_i.c"
#include "../tpccCom17/tpccCom17.h"
#include "../tpccCom17/tpccCom17_i.c"
#include "../tpccCom18/tpccCom18.h"
#include "../tpccCom18/tpccCom18_i.c"
#include "../tpccCom19/tpccCom19.h"
#include "../tpccCom19/tpccCom19_i.c"
#include "../tpccCom20/tpccCom20.h"
#include "../tpccCom20/tpccCom20_i.c"
#include "../tpccCom21/tpccCom21.h"
#include "../tpccCom21/tpccCom21_i.c"
#include "../tpccCom22/tpccCom22.h"
#include "../tpccCom22/tpccCom22_i.c"
#include "../tpccCom23/tpccCom23.h"
#include "../tpccCom23/tpccCom23_i.c"
#include "../tpccCom24/tpccCom24.h"
#include "../tpccCom24/tpccCom24_i.c"
#include "../tpccCom25/tpccCom25.h"
#include "../tpccCom25/tpccCom25_i.c"
#include "../tpccCom26/tpccCom26.h"
#include "../tpccCom26/tpccCom26_i.c"
#include "../tpccCom27/tpccCom27.h"
#include "../tpccCom27/tpccCom27_i.c"
#include "../tpccCom28/tpccCom28.h"
#include "../tpccCom28/tpccCom28_i.c"
#include "../tpccCom29/tpccCom29.h"
#include "../tpccCom29/tpccCom29_i.c"
#include "../tpccCom30/tpccCom30.h"
#include "../tpccCom30/tpccCom30_i.c"
#include "../tpccCom31/tpccCom31.h"
#include "../tpccCom31/tpccCom31_i.c"
#include "../tpccCom32/tpccCom32.h"
#include "../tpccCom32/tpccCom32_i.c"
#include "../tpccCom33/tpccCom33.h"
#include "../tpccCom33/tpccCom33_i.c"
#include "../tpccCom34/tpccCom34.h"
#include "../tpccCom34/tpccCom34_i.c"
#include "../tpccCom35/tpccCom35.h"
#include "../tpccCom35/tpccCom35_i.c"

```

```

#include "../tpccCom36/tpccCom36.h"
#include "../tpccCom36/tpccCom36_i.c"
#include "../tpccCom37/tpccCom37.h"
#include "../tpccCom37/tpccCom37_i.c"
#include "../tpccCom38/tpccCom38.h"
#include "../tpccCom38/tpccCom38_i.c"
#include "../tpccCom39/tpccCom39.h"
#include "../tpccCom39/tpccCom39_i.c"
#include "../tpccCom40/tpccCom40.h"
#include "../tpccCom40/tpccCom40_i.c"
#include "../tpccCom41/tpccCom41.h"
#include "../tpccCom41/tpccCom41_i.c"
#include "../tpccCom42/tpccCom42.h"
#include "../tpccCom42/tpccCom42_i.c"
#include "../tpccCom43/tpccCom43.h"
#include "../tpccCom43/tpccCom43_i.c"
#include "../tpccCom44/tpccCom44.h"

#include "../tpccCom44/tpccCom44_i.c"
#include "../tpccCom45/tpccCom45.h"
#include "../tpccCom45/tpccCom45_i.c"
#include "../tpccCom46/tpccCom46.h"
#include "../tpccCom46/tpccCom46_i.c"
#include "../tpccCom47/tpccCom47.h"
#include "../tpccCom47/tpccCom47_i.c"
#include "../tpccCom48/tpccCom48.h"
#include "../tpccCom48/tpccCom48_i.c"
#include "../tpccCom49/tpccCom49.h"
#include "../tpccCom49/tpccCom49_i.c"
#include "../tpccCom50/tpccCom50.h"
#include "../tpccCom50/tpccCom50_i.c"
#endif
#include "tpccOracleGlue.h"
#include <time.h>
#include <winsock.h>
#include "../tpccisapi/tpccisapi.hpp"

char tmpbuff[128];
CRITICAL_SECTION debugMutex;
CRITICAL_SECTION errorMutex;
char OracleHome[256];

#define INVALID_ITEM 100

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

void print_time_prefix(FILE *file)
{
    time_t cur_timet;
    char time_str[30];

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

    fprintf(file, "%s - ",
            time_str);
}

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 >= TIME_PREFIX_LEN) {
        fprintf(stderr, "TIME_PREFIX_LEN (%d) too small: %d\n",
                TIME_PREFIX_LEN, len);
        exit(12);
    }
    return(buffer);
}

void oralogprintf(FILE* server_logtrans, char *format, ...)
{
    char formatBuffer[200];
    char *fmt = formatBuffer;
    int fmtLen;
    va_list ap;
    va_start(ap, format);

    fmtLen = TIME_PREFIX_LEN + (int)strlen(format) + 2;
    if (fmtLen > sizeof(formatBuffer)) {
        fmt = (char *)malloc(fmtLen);
    }
    get_time_prefix(fmt);
    strcat(fmt, format);
    if (server_logtrans) {
        vfprintf(server_logtrans, fmt, ap);
        fflush(server_logtrans);
    } else {
        vfprintf(stderr, fmt, ap);
    }
    if (fmt != formatBuffer) free(fmt);
    va_end(ap);
}

/*
 * gettimeofday is not available in the Microsoft C/C++ Run Time
 * and the Win32 API.
 */

/*
 * It is not used and just for unix compatibility.
 */
struct timezone {
    char a;
};

void get_time_init();

int gettimeofday(struct timeval *curTimeP, struct timezone
*timezoneP);

#define Li2Double(x) ((double)((x).HighPart) * 4.294967296E9 +
(double)((x).LowPart))

LARGE_INTEGER pFreq;
double sFreq;

void get_time_init()
{
    QueryPerformanceFrequency(&pFreq);
    sFreq=Li2Double(pFreq);
}

void get_local_time(struct timeval *timeP)
{
    double cur_t;
    LARGE_INTEGER counter;

    QueryPerformanceCounter(&counter);
    cur_t = Li2Double(counter) / sFreq;
    timeP->tv_sec = (long)cur_t;
    timeP->tv_usec = ((long)cur_t - timeP->tv_sec) * 1000000;
}

```

```

int gettimeofday(struct timeval *curTimeP, struct timezone
*timezoneP)
{
    get_local_time(curTimeP);
    return 1;
}

BOOL APIENTRY DllMain( HANDLE hModule,
                      DWORD ul_reason_for_call,
                      LPVOID lpReserved
                      )
{
    switch (ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:

            if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTRY_SUB_
KEY,0,KEY_READ,&registryKey) == ERROR_SUCCESS)
            {
                regValueSize = 256;
                if
                (RegQueryValueEx(registryKey,"OracleHome",0,&regType,(BYTE
*)&value,&regValueSize)== ERROR_SUCCESS) {
                    strcpy(OracleHome,value);
                }
                regValueSize = sizeof(value);
                if
                (RegQueryValueEx(registryKey,"dbUserName",0,&regType,(BYTE
*)&value,&regValueSize) == ERROR_SUCCESS )
                    strcpy(userName,value);
                else
                    return ERR_INVALID_USERNAME;

                regValueSize = sizeof(value);
                if
                (RegQueryValueEx(registryKey,"dbPassword",0,&regType,(BYTE
*)&value,&regValueSize)== ERROR_SUCCESS )
                    strcpy(userPassword,value);
                else
                    return ERR_INVALID_PASSWORD;
            }
            else
            {
                return ERR_INVALID_REGISTRY_KEY;
            }
            break;
        case DLL_THREAD_ATTACH:
        case DLL_THREAD_DETACH:
        case DLL_PROCESS_DETACH:
            break;
    }
    return TRUE;
}

extern "C" TPCCORACLEGLUE_API int connect_db(char
*dbName,void **ctx)
{
    // FILE *server_logtrans;

    // server_logtrans =
fopen("C:\\inetpub\\wwwroot\\tpcc\\my_debug_gluecode.txt", "w+");
    // oralogprintf(server_logtrans,"Opening logfile\n");
    // fflush(server_logtrans);
    // fclose(server_logtrans);

    *ctx = (void *)get_db_ready((ora_cn_data_t *)ctx, userName,
userPassword, dbName);
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    if (ctx == NULL)
    {

```

```

        oralogprintf(curP->server_logtrans,"Object get_db_ready()
failed\n");

        return ERR_SAVING_CONTEXT;
    }
    return OK;
}

extern "C" TPCCORACLEGLUE_API int disconnect_db(void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    clean_connection(curP->server_logtrans, ctx);
    return OK;
}

extern "C" TPCCORACLEGLUE_API int do_nord(nord_wrapper
*nord,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    static int numCalls = 0;

    memcpy(&curP->globals,nord,sizeof(nord_wrapper));
    int rc = TPCnew((void *)curP);
    if (rc != 0 && rc != INVALID_ITEM)
    {
        int i;
        oralogprintf(curP->server_logtrans,"Error TPCnew : terror %d, rc
%d, retry %d, w_id %d, d_id %d, c_id %d, o_ol_cnt %d (out cnt
%d)\n",
        curP->globals.out_nord.terror, rc, curP-
>globals.out_nord.retry,
        curP->globals.in_nord.s_W_ID, curP-
>globals.in_nord.s_D_ID,
        curP->globals.in_nord.s_C_ID, curP-
>globals.in_nord.s_O_OL_CNT, curP-
>globals.out_nord.s_O_OL_CNT);
        for (i=0; i<15; i++)
        {
            oralogprintf(curP->server_logtrans,"ol_i_id %d,
ol_supply_w_id %d, ol_quantity %d\n",
            curP->globals.in_nord.s_OL_I_ID[i], curP-
>globals.in_nord.s_OL_SUPPLY_W_ID[i],
            curP->globals.in_nord.s_OL_QUANTITY[i]);
        }
        memcpy(nord,&curP->globals,sizeof(nord_wrapper));
        nord->out_nord.s_transtatus = rc == 0 ? TPCC_SUCCESS : curP-
>globals.out_nord.terror;
        // if (++numCalls % 10000 == 0) {
        //     oralogprintf(curP->server_logtrans,"doNewOrder so far %d\n",
numCalls);
        // }

        return OK;
    }
}

extern "C" TPCCORACLEGLUE_API int do_pymt(paym_wrapper
*paym,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    static int numCalls = 0;

    paym->in_paym.bylastname = ((paym->in_paym.s_C_ID == 0) ? 1 :
0);
    memcpy(&curP->payP,paym,sizeof(paym_wrapper));

    int rc = TPCpay((void *)curP);
    if (rc != 0)
    {
        oralogprintf(curP->server_logtrans,"Error TPCpay: terror %d, rc
%d, retry %d, w_id %d, D_id %d, C_w_id %d, c_id %d, bylastname
%d, amount %d, c_last %s (%s)\n",

```

```

        curP->payP.out_paym.terror, rc, curP->payP.out_paym.retry,
        curP->payP.in_paym.s_W_ID,
        curP->payP.in_paym.s_D_ID,
        curP->payP.in_paym.s_C_W_ID,
        curP->payP.in_paym.s_C_ID,
        curP->payP.in_paym.bylastname,
        curP->payP.in_paym.s_H_AMOUNT,
        curP->payP.in_paym.s_C_LAST ? curP-
>payP.in_paym.s_C_LAST : "-NULL-",
        (char *)pymt->out_paym.s_C_LAST ? (char *)pymt-
>out_paym.s_C_LAST : "-NULL-");
    }
    memcpy(pymt,&curP->payP,sizeof(paym_wrapper));
    pymt->out_paym.s_transtatus = rc == 0 ? TPCC_SUCCESS :
curP->payP.out_paym.terror;
// if (++numCalls % 10000 == 0) {
//     oralogprintf(curP->server_logtrans,"doPayment so far %d\n",
numCalls);
// }

    return OK;
}
extern "C" TPCCORACLEGLUE_API int do_ords(ords_wrapper
*ords,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    static int numCalls = 0;

    // oralogprintf(curP->server_logtrans,"OrderStatus, I'm tid
%d\n",ThreadNum);
    ords->in_ords.bylastname = ((ords->in_ords.s_C_ID == 0) ? 1 : 0);
    memcpy(&curP->ordP,ords,sizeof(ords_wrapper));
    int rc = TPCOrd((void *)curP);
    if (rc != 0)
    {
        oralogprintf(curP->server_logtrans,"Error TPCOrd: terror %d, rc
%d, retry %d, w_id %d, d_id %d, c_id %d, bylastname %d, c_last
%s\n",
        curP->ordP.out_ords.terror, rc, curP->ordP.out_ords.retry,
        curP->ordP.in_ords.s_W_ID, curP->ordP.in_ords.s_D_ID,
        curP->ordP.in_ords.s_C_ID,
        curP->ordP.in_ords.bylastname, curP-
>ordP.in_ords.s_C_LAST);
    }
    memcpy(ords,&curP->ordP,sizeof(ords_wrapper));
    ords->out_ords.s_transtatus = rc == 0 ? TPCC_SUCCESS : curP-
>ordP.out_ords.terror;
// if (++numCalls % 10000 == 0) {
//     oralogprintf(curP->server_logtrans,"doOrderStatus so far
%d\n", numCalls);
// }

    return OK;
}

extern "C" TPCCORACLEGLUE_API int do_dlvly(dlvly_wrapper
*dlvy,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    dlvly_wrapper *delP=&(curP->delP);
    FILE *server_logtrans = curP->server_logtrans;
    dvoid *errhp = curP->errhp;
    int rc;
    static int numCalls = 0;

    memcpy(&curP->delP,dlvy,sizeof(dlvly_wrapper));
    delP->out_dlvly.retry = 0;
    delP->in_dlvly.plsqlflag = 1;

    OCIERROR(errhp,OCIDateSysDate((OCIError *)errhp,&delP-
>in_dlvly.cr_date));

```

```

    if (rc = tkvcd (curP)) {
        if (delP->out_dlvly.terror == DEL_ERROR)
            return DEL_ERROR;
        return (-1);
    }
    memcpy(dlvly,&curP->delP,sizeof(dlvly_wrapper));
    dlvly->out_dlvly.s_transtatus = rc == 0 ? TPCC_SUCCESS : delP-
>out_dlvly.terror;
// if (++numCalls % 10000 == 0) {
//     oralogprintf(curP->server_logtrans,"doOrderStatus so far %d\n",
numCalls);
// }

    return OK;
}

extern "C" TPCCORACLEGLUE_API int do_stok(stok_wrapper
*stok,void *ctx)
{
    ora_cn_data_t *curP = (ora_cn_data_t *)ctx;
    static int numCalls = 0;
    FILE *server_logtrans = curP->server_logtrans;
    // oralogprintf(curP->server_logtrans,"StockLevel, I'm tid
%d\n",ThreadNum);

    //call stock level txn
    memcpy(&curP->stoP,stok,sizeof(stok_wrapper));
    int rc = TPCsto((void *)curP);
    if ( rc != 0 ) {
        oralogprintf(curP->server_logtrans,"Error TPCsto : terror %d, rc
%d, retry %d, w_id %d, d_id %d, threshold %d\n",
        curP->stoP.out_stok.terror, rc, curP->stoP.out_stok.retry,
        curP->stoP.in_stok.s_W_ID, curP->stoP.in_stok.s_D_ID,
        curP->stoP.in_stok.s_threshold);
    }
    memcpy(stok,&curP->stoP,sizeof(stok_wrapper));
    stok->out_stok.s_transtatus = rc == 0 ? TPCC_SUCCESS : curP-
>stoP.out_stok.terror;
// if (++numCalls % 10000 == 0) {
//     oralogprintf(curP->server_logtrans,"doStockLevel so far
%d\n", numCalls);
// }

    return OK;
}

```

tpccpl.cpp

```

#include "../stdafx.h"
#define ORA_NT
#include <stdio.h>
#include <time.h>
#include "../tpccisapi/tpcc.h"
#include "plora.h"
#include "ora_tpcc.h"
#include <stdarg.h>

extern int server_null_test;

extern CRITICAL_SECTION debugMutex;

#define INVALID_ITEM 100

#define SQLTXT "alter session set isolation_level = serializable"
#define SQLTXTTRC "alter session set sql_trace = true"
#define SQLTXTTIM "alter session set timed_statistics = true"
#define SQLTXTIMU "alter session set '\_in_memory_undo' = true"
#define SQLXTNCP "alter session set
plsql_compiler_flags=anonymous_block_native"

#ifdef ORA_NT

```

```

#undef boolean
#include "dpcbcore.h"
#define gettime dptime
#else
extern double gettime ();
#endif

static int init_cn_data(struct ora_cn_data_t *dataP);
static void initOCIhandles(struct ora_cn_data_t *cn_dataP, char*
uid, char *pwd);

extern void oralogprintf(FILE *,char *format, ...);

int proc_no;
static char *db_uid;
static char *db_pwd;

#ifdef AVOID_DEADLOCK
void swap(nord_wrapper *newP, int i, int j, int *indx, int in);
void q_sort(nord_wrapper *newP,int left, int right, int *indx, int in);
#endif

#ifdef TUX
void userlog (char* ftmp, ...)
{
    va_list va;
    va_start(va,ftmp);
    vfprintf(stderr,ftmp,va);
    va_end(va);
}
#endif

/* vmm313 void ocierror(fname, lineno, errhp, status) */
int ocierror(FILE *server_logtrans,char *fname, int lineno, dvoid
*errhp, sword status)
{
    text errbuf[512];
    sb4 errcode;
    sb4 lstat;
    ub4 recno=2;

    switch (status) {
    case OCI_SUCCESS:
        break;
    case OCI_SUCCESS_WITH_INFO:
        oralogprintf(server_logtrans,"Module %s Line %d\n", fname,
lineno);
        oralogprintf(server_logtrans,"Error -
OCI_SUCCESS_WITH_INFO\n");
        lstat = OCIErrorGet ((OCIError *)errhp, recno++, (text *) NULL,
&errcode, errbuf,
            (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
        oralogprintf(server_logtrans,"Error - %s\n", errbuf);
        break;
    case OCI_NEED_DATA:
        oralogprintf(server_logtrans,"Module %s Line %d\n", fname,
lineno);
        oralogprintf(server_logtrans,"Error - OCI_NEED_DATA\n");
        return (IRRECERR);
    case OCI_NO_DATA:
        oralogprintf(server_logtrans,"Module %s Line %d\n", fname,
lineno);
        oralogprintf(server_logtrans,"Error - OCI_NO_DATA\n");
        return (IRRECERR);
    case OCI_ERROR:
        lstat = OCIErrorGet ((OCIError *)errhp, (ub4) 1,
            (text *) NULL, &errcode, errbuf,
            (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
        if (errcode == NOT_SERIALIZABLE) return (errcode);
        if (errcode == SNAPSHOT_TOO_OLD) return (errcode);
    }
}

```

```

while (lstat != OCI_NO_DATA)
{
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname,
lineno);
    oralogprintf(server_logtrans,"Error - %s\n", errbuf);
    lstat = OCIErrorGet ((OCIError *)errhp, recno++, (text *) NULL,
&errcode, errbuf,
        (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
}
return (errcode);
/* vmm313 TPCexit(1); */
/* vmm313 exit(1); */
case OCI_INVALID_HANDLE:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname,
lineno);
    oralogprintf(server_logtrans,"Error - OCI_INVALID_HANDLE\n");
    TPCexit();
    exit(-1);
case OCI_STILL_EXECUTING:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname,
lineno);
    oralogprintf(server_logtrans,"Error - OCI_STILL_EXECUTE\n");
    return (IRRECERR);
case OCI_CONTINUE:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname,
lineno);
    oralogprintf(server_logtrans,"Error - OCI_CONTINUE\n");
    return (IRRECERR);
default:
    oralogprintf(server_logtrans,"Module %s Line %d\n", fname,
lineno);
    oralogprintf(server_logtrans,"Status - %s\n", status);
    return (IRRECERR);
}
return (RECOVER);
}

FILE *vopen(char *fnam,char *mode)
{
    FILE *fd;

#ifdef DEBUG
    fprintf(stderr, "tkvopen() fnam: %s, mode: %s\n", fnam, mode);
#endif

    fd = fopen((char *)fnam,(char *)mode);
    if (!fd){
        fprintf(stderr, "fopen on %s failed\n",fnam);
        exit(-1);
    }
    return(fd);
}

int sqlfile(FILE* server_logtrans,char *fnam, text *linebuf)
{
    FILE *fd;
    int nulpt = 0;
    char realfile[512];

    //DEBUGP((" %s: %d sqlfile() fnam: %s, linebuf: %s\n", __FILE__,
__LINE__, fnam, linebuf));

    /*
        sprintf(realfile,"%s/bench/tpc/tpcc/blocks/%s",oracle_home,fnam);
    */
    sprintf(realfile,"%s",fnam);
    if ((fd = vopen(realfile,"r")) == NULL) {
        oralogprintf(server_logtrans,"%s: %d >sqlfile vopen failed\n",
__FILE__, __LINE__);
        return(-1);
    }
}

```

```

while (fgets((char *)linebuf+nulpt, SQL_BUF_SIZE,fd))
{
    nulpt = (int)strlen((char *)linebuf);
}
oralogprintf(server_logtrans,"%s: %d <sqlfile\n", __FILE__,
__LINE__);
return(nulpt);
}

#ifdef NOT
void vgetdate (unsigned char *oradt)
{
    struct tm *loctime;
    time_t int_time;

    struct ORADATE {
        unsigned char  century;
        unsigned char  year;
        unsigned char  month;
        unsigned char  day;
        unsigned char  hour;
        unsigned char  minute;
        unsigned char  second;
    } Date;
    int century;
    int cnvrtOK;

    /* assume convert is successful */
    cnvrtOK = 1;

    /* get the current date and time as an integer */
    time( &int_time);

    /* Convert the current date and time into local time */
    loctime = localtime( &int_time);

    century = (1900+loctime->tm_year) / 100;

    Date.century = (unsigned char)(century + 100);
    if (Date.century < 119 || Date.century > 120) cnvrtOK = 0;
    Date.year = (unsigned char)(loctime->tm_year+100);
    if (Date.year < 100 || Date.year > 199) cnvrtOK = 0;
    Date.month = (unsigned char)(loctime->tm_mon + 1);
    if (Date.month < 1 || Date.month > 12) cnvrtOK = 0;
    Date.day = (unsigned char)loctime->tm_mday;
    if (Date.day < 1 || Date.day > 31) cnvrtOK = 0;
    Date.hour = (unsigned char)(loctime->tm_hour + 1);
    if (Date.hour < 1 || Date.hour > 24) cnvrtOK = 0;
    Date.minute= (unsigned char)(loctime->tm_min + 1);
    if (Date.minute < 1 || Date.minute > 60) cnvrtOK = 0;
    Date.second= (unsigned char)(loctime->tm_sec + 1);
    if (Date.second < 1 || Date.second > 60) cnvrtOK = 0;

    if (cnvrtOK)
        memcpy(oradt,&Date,7);
    else
        *oradt = '\0';

    return;
}

void cvtdmy (unsigned char *oradt, char *outdate)
{
    struct ORADATE {
        unsigned char  century;
        unsigned char  year;
        unsigned char  month;
        unsigned char  day;
        unsigned char  hour;
        unsigned char  minute;

```

```

        unsigned char  second;
    } Date;

    int day,month,year;

    memcpy(&Date,oradt,7);

    year = (Date.century-100)*100 + Date.year-100;
    month = Date.month;
    day = Date.day;
    sprintf(outdate,"%02d-%02d-%04d\0",day,month,year);

    return;
}

void cvtdmyhms (unsigned char *oradt, char *outdate)
{
    struct ORADATE {
        unsigned char  century;
        unsigned char  year;
        unsigned char  month;
        unsigned char  day;
        unsigned char  hour;
        unsigned char  minute;
        unsigned char  second;
    } Date;

    int day,month,year;
    int hour,min,sec;

    memcpy(&Date,oradt,7);

    year = (Date.century-100)*100 + Date.year-100;
    month = Date.month;
    day = Date.day;
    hour = Date.hour - 1;
    min = Date.minute - 1;
    sec = Date.second - 1;

    sprintf(outdate,"%02d-%02d-%04d %02d:%02d:%02d\0",
        day,month,year,hour,min,sec);

    return;
}
#endif

void TPCexit (void)
{
    // free(Ctpcc_com::get_cur());
}

void clean_connection(FILE* server_logtrans,void *ptr)
{
    /* free trans specific cursor handles first and later the ora handles
    */
    ora_cn_data_t *cn_dataP = (ora_cn_data_t *)ptr;

    if (cn_dataP != NULL) {
        OCIServer *tpcsrv;
        OCISession *tpcusr;
        OCIEEnv *tpcenv;
        dvoid *errhp;
        OCISvcCtx *tpcsvc;

        oralogprintf(server_logtrans, "clean_connection, Freeing OCI
handles\n");

```



```

tkvcpdone(cn_dataP);
tkvcsdone(cn_dataP);
tkvcodone(cn_dataP);
tkvcndone(cn_dataP);
/* free OCI handles */
if (tpcusr = cn_dataP->tpcusr) {
    oralogprintf(server_logtrans, "free_handles>
OCIHandleFree tpcusr\n");
    OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
}
if (tpcsvc = cn_dataP->tpcsvc) {
    oralogprintf(server_logtrans, "free_handles>
OCIHandleFree tpcsvc\n");
    OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
}
if (errhp = cn_dataP->errhp) {
    oralogprintf(server_logtrans, "free_handles>
OCIHandleFree errhp\n");
    OCIHandleFree((OCIError *)errhp,
OCI_HTYPE_ERROR);
}
if (tpcsrv = cn_dataP->tpcsrv) {
    oralogprintf(server_logtrans, "free_handles>
OCIHandleFree tpcsrv\n");
    OCIHandleFree((OCIError *)tpcsrv,
OCI_HTYPE_SERVER);
}
if (tpcenv = cn_dataP->tpcenv) {
    oralogprintf(server_logtrans, "free_handles>
OCIHandleFree tpcenv\n");
    OCIHandleFree((OCIError *)tpcenv, OCI_HTYPE_ENV);
}

    oralogprintf(server_logtrans, "free_handles> free cn_dataP\n");
}
}

```

```

static void initOCIhandles(struct ora_cn_data_t *cn_dataP, char
*uid, char *pwd, char *dbName)
{
    int tracelevel = 0; /* new define */
    OCIDate cr_date;
    text stmbuf[100];
    OCIEnv *tpcenv=NULL;
    OCIServer *tpcsrv=NULL;
    OCIError *errhp;
    OCISvcCtx *tpcsvc;
    OCISession *tpcusr;
    OCISmt *curi;

    oralogprintf(cn_dataP->server_logtrans,">> initOCIhandles uid %s
pwd %s dbName %s\n",uid,pwd,dbName);
    OCIEnvCreate (&tpcenv,OCI_OBJECT|OCI_THREADED, (dvoid
*)0, 0, 0, 0, (size_t)0, (dvoid **)0);
    if (tpcenv==NULL) {
        oralogprintf(cn_dataP->server_logtrans,"Cannot get handle to
environment, OCIEnvCreate failed\n");
    }
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv,
OCI_HTYPE_SERVER, 0, (dvoid **)0);

    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp,
OCI_HTYPE_ERROR, 0, (dvoid **)0);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc,
OCI_HTYPE_SVCCTX, 0, (dvoid **)0);
    //EnterCriticalSection(&debugMutex);
    ocierror(cn_dataP->server_logtrans, __FILE__, __LINE__, errhp,
OCIServerAttach(tpcsrv, (OCIError *)errhp, (text
*)dbName,0,OCI_DEFAULT));
}

```

```

if (tpcsrv==NULL) {
    oralogprintf(cn_dataP->server_logtrans,"Cannot get handle to
server, OCIServerAttach failed\n");
}
    oralogprintf(cn_dataP->server_logtrans,"OCIServerAttach
returned %x\n",tpcsrv);
    //LeaveCriticalSection(&debugMutex);
    OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid
*)tpcsrv, (ub4)0,OCI_ATTR_SERVER, (OCIError *)errhp);

    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr,
OCI_HTYPE_SESSION, 0, (dvoid **)0);

    OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)uid,
(ub4)4,OCI_ATTR_USERNAME, (OCIError *)errhp);

    OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid *)pwd,
(ub4)4,
        OCI_ATTR_PASSWORD, (OCIError *)errhp);
    ocierror(cn_dataP->server_logtrans, __FILE__, __LINE__, errhp,
OCISessionBegin(tpcsvc, (OCIError *)errhp, tpcusr,
OCI_CRED_RDBMS, OCI_DEFAULT));

    OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0,
OCI_ATTR_SESSION, (OCIError *)errhp);

    /* run all transaction in serializable mode */

    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid **)0);
    sprintf((char *) stmbuf, SQLTXT);
    OCISmtPrepare(curi, (OCIError *)errhp, stmbuf, strlen((char
*)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);
    ocierror(cn_dataP-
>server_logtrans, __FILE__, __LINE__, errhp, OCISmtExecute(tpcsvc
, curi, (OCIError *)errhp, 1,0,0,0,OCI_DEFAULT));
    OCIHandleFree(curi, OCI_HTYPE_STMT);

    if (getenv("USE_IMU")) {
        printf("Use in_memory_undo\n");
        OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid **)0);
        sprintf((char *) stmbuf, SQLTXTIMU);
        OCISmtPrepare(curi, (OCIError *)errhp, stmbuf, strlen((char
*)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);
        ocierror(cn_dataP-
>server_logtrans, __FILE__, __LINE__, errhp, OCISmtExecute(tpcsvc
, curi, (OCIError *)errhp, 1,0,0,0,OCI_DEFAULT));
        OCIHandleFree(curi, OCI_HTYPE_STMT);
    }

    if (getenv("USE_NCOMP")) {
        printf("Use NCOMP\n");
        OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid **)0);
        sprintf((char *) stmbuf, SQLTXTNCP);
        OCISmtPrepare(curi, (OCIError *)errhp, stmbuf, strlen((char
*)stmbuf), OCI_NTV_SYNTAX, OCI_DEFAULT);
        ocierror(cn_dataP-
>server_logtrans, __FILE__, __LINE__, errhp, OCISmtExecute(tpcsvc
, curi, (OCIError *)errhp, 1,0,0,0,OCI_DEFAULT));
        OCIHandleFree(curi, OCI_HTYPE_STMT);
    }

    if (tracelevel == 3) {
        OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT, 0,
(dvoid **)0);
        memset(stmbuf,0,100);
        sprintf((char *) stmbuf, SQLTXTTIM);
        OCISmtPrepare(curi, (OCIError *)errhp, stmbuf, strlen((char
*)stmbuf),

```

```

        OCI_NTV_SYNTAX, OCI_DEFAULT);
ocierror(cn_dataP->server_logtrans, __FILE__, __LINE__, errhp,
OCIStmtExecute(tpcsvc, curi, (OCIError
*)errhp, 1, 0, 0, 0, OCI_DEFAULT));
OCIHandleFree((dvoid *)curi, OCI_HTYPE_STMT);
}

ocierror(cn_dataP-
>server_logtrans, __FILE__, __LINE__, errhp, OCIDateSysDate((OCI
Error *)errhp, &cr_date));

/* Store the handles just initialized in the thread slot
*/
cn_dataP->tpcenv = tpcenv;
cn_dataP->tpcsrv = tpcsrv;
cn_dataP->errhp = errhp;
cn_dataP->tpcsvc = tpcsvc;
cn_dataP->tpcusr = tpcusr;
cn_dataP->curi = curi;
oralogprintf(cn_dataP->server_logtrans, "<< initOCIhandles\n");
}

/*
* init_cn_data
*   Initializes all the transactions for a single connection
*/
static int init_cn_data(ora_cn_data_t *cnP, char *uid, char *pwd, char
*dbName)
{
    int status;

    oralogprintf(cnP->server_logtrans, ">> init_cn_data\n");
    initOCIhandles(cnP, uid, pwd, dbName);

    if (status = tkvcninit (cnP)) {
        fprintf(stderr, "tkvcninit failed: %d\n", status);
        TPCexit ();
        return (status);
    }

    if (status = tkvcpinit (cnP)) {
        fprintf(stderr, "tkvcpinit failed: %d\n", status);
        TPCexit ();
        return (status);
    }

    if (status = tkvcoint (cnP)) {
        fprintf(stderr, "tkvcoint failed: %d\n", status);
        TPCexit ();
        return (status);
    }

    if (status = tkvcdinit (cnP)) {
        fprintf(stderr, "tkvcdinit failed: %d\n", status);
        TPCexit ();
        return (status);
    }

    if (status = tkvcsinit (cnP)) {
        fprintf(stderr, "tkvcsinit failed: %d\n", status);
        TPCexit ();
        return (status);
    }
}
return 0;
}

void *create_ora_connection(char *uid, char *pwd, char *dbName) {
    HKEY Key;
    char value[256];
    DWORD regValueSize = 256;

```

```

    DWORD regType;
    FILE *server_logtrans;

    ora_cn_data_t *cnP = (ora_cn_data_t
*)malloc(sizeof(ora_cn_data_t));

    if
(RegOpenKeyEx(HKEY_LOCAL_MACHINE, "SOFTWARE\\TPCC",
0, KEY_READ, &Key) == ERROR_SUCCESS)
    {
        regValueSize = 256;
        if
(RegQueryValueEx(Key, "OraComServLog", 0, &regType, (BYTE
*)&value, &regValueSize) == ERROR_SUCCESS)
        {
            server_logtrans = fopen(value, "w+");
            oralogprintf(server_logtrans, "Opening logfile %s\n", value);
        }
    }
    cnP->server_logtrans = server_logtrans;
    oralogprintf(server_logtrans, ">> create_ora_connection\n");
    if (cnP!=NULL)
        init_cn_data(cnP, uid, pwd, dbName);
    else
        oralogprintf(server_logtrans, "Cannot allocate ora_cn_data_t.\n");
    return (void *)cnP;
}

ora_cn_data_t *get_db_ready(ora_cn_data_t *curP, char *uid, char
*pwd, char *dbName)
{
    curP = (ora_cn_data_t
*)create_ora_connection(uid, pwd, dbName);
    if (curP==NULL) {
        oralogprintf(curP->server_logtrans, "create_ora_connection
failed!\n");
    }

    return(curP);
}

TPCnew (void *cnP)
{
    int i;
    ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
    FILE *server_logtrans = cn_dataP->server_logtrans;
    nord_wrapper *newP = &cn_dataP->globals;
    int indx[NITEMS], ordl_cnt=newP->out_nord.s_O_OL_CNT;
    dvoid *errhp = cn_dataP->errhp;

    newP->out_nord.retry = 0;

#ifdef AVOID_DEADLOCK
    for (i = NITEMS; i > 0; i--) {
        if (newP->in_nord.s_OL_I_ID[i-1] > 0) {
            ordl_cnt = i;
            break;
        }
    }

    for (i = 0; i < NITEMS; i++) indx[i] = i;

    q_sort(newP, 0, ordl_cnt-1, indx, 1);
#endif

    OCIERROR(errhp, OCIDateSysDate((OCIError *)errhp, &newP-
>in_nord.cr_date));

```

```

if (tkvcn (cn_dataP)) {
    if (newP->out_nord.terror != RECOVERR)
    {
        if (newP->out_nord.status) {
            return(newP->out_nord.terror);
        }
        else {
            newP->out_nord.terror = INVALID_STATUS;
            return (INVALID_STATUS);
        }
    }
}

newP->out_nord.datelen = sizeof(newP-
>out_nord.s_O_ENTRY_D_time);
OCIERROR(errhp,
    OCIDateToText((OCIError *)errhp,&newP-
>in_nord.cr_date,(text*)FULLDATE,SIZ(FULLDATE),(text*)0,0,
    (ub4 *)&newP->out_nord.datelen,(text *)newP-
>out_nord.s_O_ENTRY_D_time));

#ifdef AVOID_DEADLOCK
    q_sort(newP, 0, ordl_cnt-1, indx, 0);
#endif
newP->out_nord.terror = 0;
return (0);
}

TPCpay (void *cnP)
{
    ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
    FILE *server_logtrans = cn_dataP->server_logtrans;
    paym_wrapper *payP = &cn_dataP->payP;
    dvoid *errhp = cn_dataP->errhp;
    char months[13][4] = {"",
        "JAN","FEB","MAR","APR","MAY","JUN","JUL","AUG","SEP","OCT",
        "NOV","DEC"};

    OCIERROR(errhp,OCIDateSysDate((OCIError *)errhp,&payP-
>in_paym.cr_date));

    sprintf(payP->out_paym.s_H_DATE_time,"%d-%s-%d
%d:%d:%d",payP->in_paym.cr_date.OCIDateDD,
    months[payP->in_paym.cr_date.OCIDateMM], payP-
>in_paym.cr_date.OCIDateYYYY,
    payP->in_paym.cr_date.OCIDateTime.OCITimeHH, payP-
>in_paym.cr_date.OCIDateTime.OCITimeMI,
    payP->in_paym.cr_date.OCIDateTime.OCITimeSS);

    if (payP->in_paym.bylastname) {
        payP->in_paym.s_C_ID = 0;
    }
    else {
        payP->in_paym.s_C_LAST[0] = ' ';
    }

    if (tkvcp (cn_dataP)) {
        if (payP->out_paym.terror != RECOVERR)
            payP->out_paym.terror = IRRECERR;
        return (-1);
    }

    sprintf(payP->out_paym.c_since_d,"%d-%s-%d",payP-
>out_paym.c_since.OCIDateDD,months[payP-
>out_paym.c_since.OCIDateMM],
    payP->out_paym.c_since.OCIDateYYYY);

    payP->out_paym.terror = 0;

```

```

return (0);
}

TPCord (void *cnP)
{
    ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
    ords_wrapper *ordP = &cn_dataP->ordP;

    if (tkvco (cn_dataP)) {
        if (ordP->out_ords.terror != RECOVERR)
            ordP->out_ords.terror = IRRECERR;
        return (-1);
    }

    ordP->out_ords.terror = 0;
    if ( ordP->out_ords.s_O_CARRIER_ID == 11 )
        ordP->out_ords.s_O_CARRIER_ID = 0;

    return (0);
}

int TPCdel (void *cnP)
{
    ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
    FILE *server_logtrans = cn_dataP->server_logtrans;
    dlvy_wrapper *str = &cn_dataP->delP;
    dvoid *errhp = cn_dataP->errhp;
    OCIDate cr_date;

    OCIERROR(errhp,OCIDateSysDate((OCIError *)errhp,&cr_date));

    if (tkvcd (cn_dataP)) {
        if (str->out_dlvy.terror == DEL_ERROR)
            return DEL_ERROR;
        if (str->out_dlvy.terror != RECOVERR)
            str->out_dlvy.terror = IRRECERR;
        return (-1);
    }

    str->out_dlvy.terror = 0;
    return (0);
}

TPCsto (void *cnP)
{
    ora_cn_data_t *cn_dataP = (ora_cn_data_t *)cnP;
    stok_wrapper *stoP = &cn_dataP->stoP;

    stoP->out_stok.retry = 0;

    if (tkvcs (cn_dataP)) {
        if (stoP->out_stok.terror != RECOVERR)
            stoP->out_stok.terror = IRRECERR;
        return (-1);
    }

    stoP->out_stok.terror = 0;

    return (0);
}

#ifdef AVOID_DEADLOCK
void q_sort(nord_wrapper *newP,int left, int right, int *indx, int in)
{
    int i, last;

```

```

if(left >= right)
    return;
swap(newP,left,(left+right)/2, indx, in);
last = left;
for(i=left+1;i<=right;i++)
    if(newP->in_nord.s_OL_I_ID[i] < newP->in_nord.s_OL_I_ID[left])
        swap(newP,last,i, indx, in);
swap(newP, left,last, indx, in);
q_sort(newP,left,last-1, indx, in);
q_sort(newP,last+1,right, indx, in);
}

void swap(struct nord_wrapper *newP, int i, int j, int *indx, int in)
{
    int tempd;
    int tempi;
    char tmpstr[25];
    char tmpch;

    tempi = indx[i];
    indx[i] = indx[j];
    indx[j] = tempi;

    if (in) {

        tempi = newP->in_nord.s_OL_I_ID[i];
        newP->in_nord.s_OL_I_ID[i] = newP->in_nord.s_OL_I_ID[j];
        newP->in_nord.s_OL_I_ID[j] = tempi;
        tempi = newP->in_nord.s_OL_SUPPLY_W_ID[i];
        newP->in_nord.s_OL_SUPPLY_W_ID[i] = newP->
in_nord.s_OL_SUPPLY_W_ID[j];
        newP->in_nord.s_OL_SUPPLY_W_ID[j] = tempi;

        tempi = newP->in_nord.s_OL_QUANTITY[i];
        newP->in_nord.s_OL_QUANTITY[i] = newP->
in_nord.s_OL_QUANTITY[j];
        newP->in_nord.s_OL_QUANTITY[j] = (short)tempi;
    } else {
        strcpy(tmpstr,newP->out_nord.s_I_NAME[i]);
        strcpy(newP->out_nord.s_I_NAME[i],newP->
out_nord.s_I_NAME[j]);
        strcpy(newP->out_nord.s_I_NAME[j],tmpstr);

        tempi = newP->out_nord.s_S_QUANTITY[i];
        newP->out_nord.s_S_QUANTITY[i] = newP->
out_nord.s_S_QUANTITY[j];
        newP->out_nord.s_S_QUANTITY[j] = (short)tempi;

        tmpch = newP->out_nord.s_brand_generic[i];
        newP->out_nord.s_brand_generic[i] = newP->
out_nord.s_brand_generic[j];
        newP->out_nord.s_brand_generic[j] = tmpch;

        tempd = newP->out_nord.s_I_PRICE[i];
        newP->out_nord.s_I_PRICE[i] = newP->
out_nord.s_I_PRICE[j];
        newP->out_nord.s_I_PRICE[j] = tempd;

        tempd = newP->out_nord.s_OL_AMOUNT[i];
        newP->out_nord.s_OL_AMOUNT[i] = newP->
out_nord.s_OL_AMOUNT[j];
        newP->out_nord.s_OL_AMOUNT[j] = tempd;
    }
}

#endif

ora_tpcc.h
/*

```

```

* $Header: tpcc.h 7030100.1 95/07/19 15:10:55 plai Generic<base>
$ Copyr (c) 1993 Oracle
*/
/*=====
=====+
|      Copyright (c) 1995  Oracle Corp, Redwood Shores, CA      |
|      OPEN SYSTEMS PERFORMANCE GROUP                          |
|                                                                |
|      All Rights Reserved                                     |
|                                                                |
+=====
=====+
| FILENAME
|   tpcc.h
| DESCRIPTION
|   Include file for TPC-C benchmark programs.
+=====
=====*/

#ifndef TPCC_H
#define TPCC_H

#ifndef FALSE
# define FALSE 0
#endif

#ifndef TRUE
# define TRUE 1
#endif

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

#ifndef boolean
#define boolean int
#endif

#include <tpccflags.h>

#include <oratypes.h>
#include <oci.h>
#include <ocidfn.h>
/*
#ifdef __STDC__
#include "ociapr.h"
#else
#include "ocikpr.h"
#endif
*/

typedef struct cda_def csrdef;
typedef struct cda_def ldadef;

typedef __int16 int16_t;
typedef __int32 int32_t;
typedef __int64 int64_t;

/* TPC-C transaction functions */

extern int TPCinit ();
extern int TPCnew ();
extern int TPCpay ();
extern int TPCord ();
extern int TPCdel ();
extern int TPCsto ();
extern void TPCexit ();
extern int TPCdumpinit ();

```

```

extern void TPCdumpnew ();
extern void TPCdumppay ();
extern void TPCdumpord ();
extern void TPCdumpdel ();
extern void TPCdumpsto ();
extern void TPCdumpexit ();
extern void userlog(char* fntp, ...);

/* Error codes */

#define RECOVER -10
#define IRECCERR -20
#define NOERR 111
#define DEL_ERROR -666
#define DEL_DATE_LEN 7
#define NDISTS 10
#define NITEMS 15
#define SQL_BUF_SIZE 8192

#define FULLDATE "dd-mon-yy.hh24:mi:ss"
#define SHORTDATE "dd-mm-yyyy"

#define DELRT 80.0

extern int tkvcss (); /* for alter session to get memory size and trace
*/
extern boolean multitranx;
extern int ord_init;

extern void errrpt ();
extern int ocierror(FILE *server_logtrans, char *fname, int
lineno, dvoid *errhp, sword status);
//extern int sqlfile(char *fname, text *linebuf);

extern FILE *lfp;
//extern FILE *fopen ();
extern int proc_no;
extern int doid[];

/* Miscellaneous */

#ifndef DISCARD
# define DISCARD (void)
#endif

#ifndef sword
# define sword int
#endif

#define VER7 2

#define NA -1 /* ANSI SQL NULL */
#define NLT 1 /* length for string null terminator */
#define DEADLOCK 60 /* ORA-00060: deadlock */
#define NO_DATA_FOUND 1403 /* ORA-01403: no data found
*/
#define NOT_SERIALIZABLE 8177 /* ORA-08177: transaction not
serializable */
#define SNAPSHOT_TOO_OLD 1555 /* ORA-01555: snapshot too
old */

#ifndef NULLP
# define NULLP(x) (x *)NULL
#endif /* NULLP */

#define ADR(object) ((ub1 *)&(object))
#define SIZ(object) ((sword)sizeof(object))

typedef char date[24+NLT];

```

```

typedef char varchar2;

#define min(x,y) (((x) < (y)) ? (x) : (y))

#define OCIERROR(errp,function)\
ocierror(server_logtrans,__FILE__,__LINE__,(dvoid
*)(errp),(function));

#define OCIBND(stmp, bndp, errp, sqlvar, progv, progvl, ftype)\
ocierror(server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp), &(bndp), (errp), \
(text *) (sqlvar), strlen((sqlvar)), \
(progv), (progvl), (ftype), 0, 0, 0, 0, OCI_DEFAULT));

/* bind arrays for sql */
#define
OCIBNDRA(stmp, bndp, errp, sqlvar, progv, progvl, ftype, indp, alen, arco
de) \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp), &(bndp), (errp), (text
*)(sqlvar), strlen((sqlvar)), \

(progv), (progvl), (ftype), (indp), (alen), (arcode), 0, 0, OCI_DEFAULT));

/* use with callback data */
#define OCIBNDRAD(stmp, bndp, errp, sqlvar, progvl, ftype, indp, ctxp, \
cbf_nodata, cbf_data) \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp), &(bndp), (errp), (text *) (sqlvar), \
strlen((sqlvar)), 0, (progvl), (ftype), \
indp, 0, 0, 0, 0, OCI_DATA_AT_EXEC)); \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \

OCIBindDynamic((bndp), (errp), (ctxp), (cbf_nodata), (ctxp), (cbf_data))
);

/* bind in/out for plsql without indicator and rcode */
#define OCIBNDPL(stmp, bndp, errp, sqlvar, progv, progvl, ftype, alen) \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp), &(bndp), (errp), (CONST text *) (sqlvar), \
(sb4)strlen((CONST char *) (sqlvar)),
(dvoid*) (progv), (progvl), (ftype), \
NULLP(dvoid), (alen), NULLP(ub2),
0, NULLP(ub4), OCI_DEFAULT));

/* bind in values for plsql with indicator and rcode */
#define
OCIBNDR(stmp, bndp, errp, sqlvar, progv, progvl, ftype, indp, alen, arco
de) \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp), &(bndp), (errp), (text
*)(sqlvar), strlen((sqlvar)), \

```

```

        (progv),(progl),(ftype),(indp),(alen),(arcode),0,0, \
        OCI_DEFAULT));

/* bind in/out for plsql arrays without indicator and rcode */
#define
OCIBNDPLA(stmp,bndp,err,sqlvar,progv,progl,ftype,alen,ms,cu) \
    DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(err), \
    \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0));\
    DISCARD ocierror(server_logtrans,__FILE__,__LINE__,(err),\
    OCIBindByName((stmp),&(bndp),(err),(CONST text
*)(sqlvar), \
        (sb4)strlen((CONST char *) (sqlvar)),(void *) (progv), \

(progl),(ftype),NULL,(alen),NULL,(ms),(cu),OCI_DEFAULT));

/* bind in/out values for plsql with indicator and rcode */
#define
OCIBNDRAA(stmp,bndp,err,sqlvar,progv,progl,ftype,indp,alen,arc
ode,\
    ms,cu) \
    ocierror(server_logtrans,__FILE__,__LINE__,(err), \

OCIHandleAlloc((stmp),(dvoid**)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0));\
    ocierror(server_logtrans,__FILE__,__LINE__,(err),\
    OCIBindByName((stmp),&(bndp),(err),(text
*)(sqlvar),strlen((sqlvar)),\

(progv),(progl),(ftype),(indp),(alen),(arcode),(ms),(cu),OCI_DEFAU
LT));

#define OCIDEFINE(stmp,dfnp,err,pos,progv,progl,ftype)\

OCIDefineByPos((stmp),&(dfnp),(err),(pos),(progv),(progl),(ftype),\
    0,0,0,OCI_DEFAULT);

#define OCIDEF(stmp,dfnp,err,pos,progv,progl,ftype) \

OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
    (dvoid**0));\
    OCIDefineByPos((stmp),&(dfnp),(err),(pos),(progv),(progl),\
        (ftype),NULL,NULL,NULL,OCI_DEFAULT); \

#define
OCIDFNRA(stmp,dfnp,err,pos,progv,progl,ftype,indp,alen,arcode)
\

OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
    (dvoid**0));\
    OCIDefineByPos((stmp),&(dfnp),(err),(pos),(progv),\
        (progl),(ftype),(indp),(alen),\
        (arcode),OCI_DEFAULT);

#define
OCIDFNNDYN(stmp,dfnp,err,pos,progv,progl,ftype,indp,ctxp,cbf_d
ata) \
    ocierror(server_logtrans,__FILE__,__LINE__,(err), \

OCIHandleAlloc((stmp),(dvoid**)&(dfnp),OCI_HTYPE_DEFINE,0,\
    (dvoid**0));\
    ocierror(server_logtrans,__FILE__,__LINE__,(err), \
    OCIDefineByPos((stmp),&(dfnp),(err),(pos),(progv),\
        (progl),(ftype),\
            (indp),NULL,NULL,
OCI_DYNAMIC_FETCH));\
    ocierror(server_logtrans,__FILE__,__LINE__,(err), \
    OCIDefineDynamic((dfnp),(err),(ctxp),(cbf_data));

```

```

struct out_stocklev_struct {
    int32_t terror;
    int32_t s_low_stock;
    int32_t retry;
    int16_t s_transtatus;
};

struct in_stocklev_struct {
    int32_t s_threshold;
    int32_t s_D_ID;
    int32_t s_W_ID;
};

struct in_neword_struct {
    int32_t s_OL_I_ID[15];
    int32_t s_OL_SUPPLY_W_ID[15];
    int32_t s_OL_QUANTITY[15];
    int32_t s_C_ID;
    int32_t s_W_ID;
    int32_t s_D_ID;
    int32_t s_O_OL_CNT;
    int32_t s_all_local;
    int32_t duplicate_items;
    OCIDate cr_date;
};

struct out_neword_struct {
    int32_t s_I_PRICE[15];
    int32_t s_OL_AMOUNT[15];
    int32_t s_S_QUANTITY[15];
    char s_I_NAME[15][25];
    char s_brand_generic[15];
    char s_O_ENTRY_D_time[22];
    float s_W_TAX;
    float s_D_TAX;
    float s_C_DISCOUNT;
    float s_total_amount;
    int32_t s_O_ID;
    int32_t s_O_OL_CNT;
    int16_t s_transtatus;
    int16_t deadlocks;
    int16_t datelen;
    char s_C_LAST[17];
    char s_C_CREDIT[3];
    int32_t terror;
    int32_t status;
    int32_t retry;
    int16_t items_valid;
};

struct in_payment_struct {
    int s_H_AMOUNT;
    int32_t s_W_ID;
    int32_t s_C_W_ID;
    int s_C_ID;
    int32_t s_C_D_ID;
    int32_t s_D_ID;
    char s_C_LAST[17];
    int32_t bylastname;
    OCIDate cr_date;
};

struct out_payment_struct {
    char s_H_DATE_time[22];
    OCIDate c_since;
    char c_since_d[12];
    double s_C_CREDIT_LIM;
    double s_C_BALANCE;
    float s_C_DISCOUNT;
    int s_C_ID;
};

```

```

int16_t s_transtatus;
int16_t since_len;
int32_t terror;
int16_t hlen;
int32_t retry;
char s_W_STREET_1[21];
char s_W_STREET_2[21];
char s_W_CITY[21];
char s_W_STATE[3];
char s_W_ZIP[10];
char s_D_STREET_1[21];
char s_D_STREET_2[21];
char s_D_CITY[21];
char s_D_STATE[3];
char s_D_ZIP[10];
char s_C_FIRST[17];
char s_C_MIDDLE[3];
char s_C_LAST[17];
char s_C_STREET_1[21];
char s_C_STREET_2[21];
char s_C_CITY[21];
char s_C_STATE[3];
char s_C_ZIP[10];
char s_C_PHONE[17];
char s_C_CREDIT[3];
char s_C_DATA[201];
};

struct in_ordstat_struct {
int32_t s_C_ID;
int32_t s_W_ID;
int32_t s_D_ID;
int32_t bylastname;
char s_C_LAST[17];
OCIDate cr_date;
};

struct out_ordstat_struct {
double s_C_BALANCE;
char s_O_ENTRY_D_time[22];
int32_t s_C_ID;
int32_t s_O_ID;
int32_t s_O_CARRIER_ID;
int32_t s_ol_cnt;
int32_t terror;
char s_OL_DELIVERY_D_time[15][22];
OCIDate s_OL_DELIVERY_D_base[15];
int32_t s_OL_AMOUNT[15];
int32_t s_OL_I_ID[15];
int32_t s_OL_SUPPLY_W_ID[15];
int32_t s_OL_QUANTITY[15];
int32_t ol_del_len[15];
int16_t s_transtatus;
int32_t retry;
char s_C_FIRST[17];
char s_C_MIDDLE[3];
char s_C_LAST[17];
};

struct in_delivery_struct {
double s_O_DELIVERY_D_time;
int32_t s_W_ID;
int32_t s_O_CARRIER_ID;
int16_t plsflag;
OCIDate cr_date;
};

struct out_delivery_struct {
int32_t s_O_ID[10];
int32_t terror;
int16_t s_transtatus;
int32_t retry;
};

```

```

};

#endif

oratapcc.h

#ifndef ORATPCC_H
#define ORATPCC_H

typedef struct {
long int sec;
long int usec;
} time_type;

typedef struct {
OCIDate cr_date;
int dtype;
int returncode;
long int sql_code;
long int isam_code;
long int num_rms;

short int stats; /* For instrument only */
time_type start_time; /* For Debug Purposes only */
time_type end_time; /* For Debug Purposes only */
} data_header;

struct stoinstruct {
int w_id;
int d_id;
#ifdef USE_IEEE_NUMBER
float threshold;
#else
int threshold;
#endif
};

struct stooutstruct {
int terror;
int low_stock;
int retry;
};

struct stostruct {
data_header header;
struct stoinstruct stoin;
struct stooutstruct stoout;
int out_s_transtatus;
};

struct newinstruct {
int w_id;
int d_id;
int c_id;
int ol_i_id[15];
int ol_supply_w_id[15];
#ifdef USE_IEEE_NUMBER
float ol_quantity[15];
#else
int ol_quantity[15];
#endif
};

struct newoutstruct {
int terror;
int o_id;
int o_ol_cnt;
char c_last[17];
char c_credit[3];
float c_discount;
float w_tax;
float d_tax;
char o_entry_d[20];
};

```

```

    int datelen;
#ifdef USE_IEEE_NUMBER
    float i_price[15];
    float ol_amount[15];
    float s_quantity[15];
#else
    int i_price[15];
    int ol_amount[15];
    int s_quantity[15];
#endif
    float total_amount;
    char i_name[15][25];
    char brand_generic[15];
    int status;
    int retry;
    int o_all_local;
};

struct newstruct {
    data_header header;
    struct newinstruct newin;
    struct newoutstruct newout;
    int out_s_transtatus;
    int items_valid;
};

struct payinstruct {
    int w_id;
    int d_id;
    int c_w_id;
    int c_d_id;
    int c_id;
    int bylastname;
#ifdef USE_IEEE_NUMBER
    float h_amount;
#else
    int h_amount;
#endif
    char c_last[17];
};

struct payoutstruct {
    int terror;
    int hlen;
    char w_street_1[21];
    char w_street_2[21];
    char w_city[21];
    char w_state[3];
    char w_zip[10];
    char d_street_1[21];
    char d_street_2[21];
    char d_city[21];
    char d_state[3];
    char d_zip[10];
    int c_id;
    char c_first[17];
    char c_middle[3];
    char c_last[17];
    char c_street_1[21];
    char c_street_2[21];
    char c_city[21];
    char c_state[3];
    char c_zip[10];
    char c_phone[17];
    OCIDate c_since;
    char c_since_d[11];
    int since_len;
    char c_credit[3];
    double c_credit_lim;
    float c_discount;
    double c_balance;
    char c_data[201];

```

```

    char h_date[20];
    int retry;
};

struct paystruct {
    data_header header;
    struct payinstruct payin;
    struct payoutstruct payout;
    int out_s_transtatus;
};

struct ordinstruct {
    int w_id;
    int d_id;
    int c_id;
    int bylastname;
    char c_last[17];
};

struct ordoutstruct {
    int terror;
    int c_id;
    char c_last[17];
    char c_first[17];
    char c_middle[3];
    double c_balance;
    int o_id;
    OCIDate o_entry_d_base;
    OCIDate ol_d_base[15];
    char o_entry_d[20];
    int o_carrier_id;
    int o_ol_cnt;
    int ol_supply_w_id[15];
    int ol_i_id[15];
#ifdef USE_IEEE_NUMBER
    float ol_quantity[15];
    float ol_amount[15];
#else
    int ol_quantity[15];
    int ol_amount[15];
#endif
    char ol_delivery_d[15][11];
    int ol_del_len[15];
    int retry;
};

struct ordstruct {
    data_header header;
    struct ordinstruct ordin;
    struct ordoutstruct ordout;
    int out_s_transtatus;
};

struct delinstruct {
    int w_id;
    int o_carrier_id;
    double qtime;
    int in_timing_int;
    int plsflag;
};

struct deloutstruct {
    int del_o_id[10];
    int terror;
    int retry;
};

struct delstruct {
    data_header header;
    struct delinstruct delin;
    struct deloutstruct delout;
    OCIDate cr_date;

```



```

int out_s_transtatus;
};
#endif /* ORATPCC_H */

```

plora.h

```

/*
 * $Header: plora.h for Encina
 */
/*=====+
|           All Rights Reserved           |
+=====+
| FILENAME
|   plora.h
| DESCRIPTION
|   Include file for TPC-C benchmark programs.
+=====+
=====*/

#ifndef TPCC_PLORA_H
#define TPCC_PLORA_H

#define serverDebug 1
//extern void logprintf(char *format, ...);
//define DEBUGP(list) if (serverDebug) logprintf list
#define TIME_PREFIX_LEN 50
//extern FILE *server_logtrans;

#define NEWO_TRANS (1)
#define PAYMENT_TRANS (2)
#define ORDER_STAT_TRANS (3)
#define DELIVERY_TRANS (4)
#define STOCK_TRANS (5)
#define MAX_TRAN_TYPE (5)

/* Oracle handles and rest of thread specific vars(thread slot data ) */

struct ora_cn_data_t {
    OCIEnc *tpcenv;
    OCIServer *tpcsrv;
    dvoid *errhp;
    OCISvcCtx *tpcsvc;
    OCISession *tpcusr;
    OCISmt *curi;
    dvoid *xmem;

    nord_wrapper globals;
    ords_wrapper ordP;
    stok_wrapper stoP;
    paym_wrapper payP;
    dlvy_wrapper delP;

    int tid;
    FILE* server_logtrans;

    void *nctx;
    void *pctx;
    void *octx;
    void *sctx;
    void *dctx;
    void *pldctx;
    void *actx; /* for #ifdef DMLRETDL */
    void *cbctx; /* for orderstatus */
    void *ctxp_octx; /* for orderstatus */
};

```

```

struct thread_info
{
    ora_cn_data_t *curP;
    FILE *server_logtrans;
    int tid;
    int server_null_test;
    char oracle_home[255];
    char dbName[20];
};

```

```

typedef struct ora_cn_data_t ora_cn_data_t;
extern int tkvcninit (ora_cn_data_t *ora_SlotDataP);
extern int tkvcpinit (ora_cn_data_t *ora_SlotDataP);
extern int tkvcoint (ora_cn_data_t *ora_SlotDataP);
extern int tkvcdinit (ora_cn_data_t *ora_SlotDataP);
extern int tkvcsinit (ora_cn_data_t *ora_SlotDataP);

```

```

extern int tkvcn (ora_cn_data_t *ora_SlotDataP);
extern int tkvcp (ora_cn_data_t *ora_SlotDataP);
extern int tkvco (ora_cn_data_t *ora_SlotDataP);
extern int tkvcd (ora_cn_data_t *ora_SlotDataP);
extern int tkvcs (ora_cn_data_t *ora_SlotDataP);

```

```

extern void tkvcndone (ora_cn_data_t *ora_SlotDataP);
extern void tkvcpdone (ora_cn_data_t *ora_SlotDataP);
extern void tkvcodone (ora_cn_data_t *ora_SlotDataP);
extern void tkvcddone (ora_cn_data_t *ora_SlotDataP);
extern void tkvcsdone (ora_cn_data_t *ora_SlotDataP);

```

```

#endif /* TPCC_PLORA_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

```

```

#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

```

tpccflags.h

```

#define PLSQLNO
#define DMLRETDL

```

tpcc_info.h

```

/*
 * $Header: tpcc_info.h 7030100.1 95/07/19 15:11:37 plai
Generic<base> $ Copyr (c) 1995 Oracle
 */
/*=====+
|           Copyright (c) 1995 Oracle Corp, Redwood Shores, CA           |
|           OPEN SYSTEMS PERFORMANCE GROUP                               |
|
|           All Rights Reserved           |
+=====+
| FILENAME
|   tpcc_info.h
| DESCRIPTION
|   Include file for TPC-C benchmark programs.

```

```

+=====
=====*/

#ifndef TPCC_INFO_H
#define TPCC_INFO_H

/* this set is duplicated in c_Defs.h, c_Defs.h is used for batch driver
*/
#define MENTXN 0 /* menu txn */
#define NEWTXN 1 /* new order transaction */
#define PAYTXN 2 /* payment transaction */
#define ORDTXN 3 /* order status transaction */
#define DELTXN 4 /* delivery transaction */
#define STOTXN 5 /* stock level transaction */
#define ALLTXN 6 /* for processing all txns */
#define ALLTXNMODEL 7 /* for processing all txns except
delivery */
#endif

tpccOracleGlue.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
TPCCORACLEGLUE_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
// TPCCORACLEGLUE_API functions as being imported from a
DLL, whereas this DLL sees symbols
// defined with this macro as being exported.
#ifdef TPCCORACLEGLUE_EXPORTS
#define TPCCORACLEGLUE_API __declspec(dllexport)
#else
#define TPCCORACLEGLUE_API __declspec(dllimport)
#endif

#include "../tpccisapi/tpcc.h"
#include "ora10_mt/ora_tpcc.h"
#include "ora10_mt/plora.h"

////////////////////////////////////
// Error/Debug log file defines
////////////////////////////////////
ofstream debugStream;
ofstream errorStream;

////////////////////////////////////
// Registry Values
////////////////////////////////////
#define DB_USER_NAME "dbUserName"
#define DB_USER_PASSWORD "dbPassword"
#define DB_NAME "dbName"

char userName[16] = {NULL};
char userPassword[16] = {NULL};

HKEY registryKey;
DWORD regType;
char value[MAX_STRING_LEN];
DWORD regValueSize = MAX_STRING_LEN;

////////////////////////////////////
// Oracle Glue Function Prototypes
////////////////////////////////////
extern "C" TPCCORACLEGLUE_API int connect_db(char
*dbName,void **ctx);
extern "C" TPCCORACLEGLUE_API int getContext(void **ctx);
extern "C" TPCCORACLEGLUE_API int detachContext(void *ctx);
extern "C" TPCCORACLEGLUE_API int attachContext(void *ctx);

```

```

extern "C" TPCCORACLEGLUE_API int disconnect_db(void *ctx);

extern "C" TPCCORACLEGLUE_API int do_nord(nord_wrapper
*nord,void *ctx);
extern "C" TPCCORACLEGLUE_API int do_pymt(paym_wrapper
*pymt,void *ctx);
extern "C" TPCCORACLEGLUE_API int do_ordr(ordr_wrapper
*ordr,void *ctx);
extern "C" TPCCORACLEGLUE_API int do_dlv(dlv_wrapper
*dlv,void *ctx);
extern "C" TPCCORACLEGLUE_API int do_stok(stok_wrapper
*stok,void *ctx);

////////////////////////////////////
// Function Prototypes
////////////////////////////////////
extern ora_cn_data_t *get_db_ready(ora_cn_data_t *ctx, char *uid,
char *pwd, char *dbName);
extern void clean_connection(FILE* server_logtrans,void *ptr);

extern int TPCnew (void *cnP);
extern int TPCpay (void *cnP);
extern int TPCord (void *cnP);
extern int TPCsto (void *cnP);
extern int TPCdel (void *cnP);

extern CRITICAL_SECTION debugMutex;

#define TPCC_SUCCESS 0

// This class is exported from the tpccOracleGlue.dll
class TPCCORACLEGLUE_API CtpccOracleGlue {
public:
    CtpccOracleGlue(void);
    // TODO: add your methods here.
};

tpccpl.h

/*
* $Header: tpccpl.h 7030100.1 96/04/02 18:03:35 plai
Generic<base> $ Copyr (c) 1994 Oracle
*/
/*=====
=====+
| Copyright (c) 1994 Oracle Corp, Redwood Shores, CA |
| OPEN SYSTEMS PERFORMANCE GROUP |
| All Rights Reserved |
+=====
=====+
| FILENAME |
| tpccpl.h |
| DESCRIPTION |
| Header file for TPC-C transactions in PL/SQL. |
+=====
=====*/

#ifndef TPCCPL_H
#define TPCCPL_H

#include <stdio.h>
#include "tpcc.h"

#ifdef TUX
#define DELRT 5.0
#else
#define DELRT 80.0
#endif

```

```

extern int plnewinit ();
extern int plpayinit ();
extern int plordinit ();
extern int pldelinit ();
extern int plstoinit ();

extern int plnew ();
extern int plpay ();
extern int plord ();
extern int pldel ();
extern int plsto ();

extern void plnewdone ();
extern void plpaydone ();
extern void plorddone ();
extern void pldelone ();
extern void plstodone ();

extern errrpt ();
extern void logerr();
extern int ocierror(FILE *server_logtrans,char *fname, int
lineno,OCIError *errhp, sword status);

extern int sqlfile(char *fname, text *linebuf);

extern void cvtdmy ( unsigned char *orady, char *outdate);
extern void cvtdmyhms (unsigned char *orady, char *outdate);

extern FILE *fip;
extern FILE *fopen ();
extern int proc_no;
extern int doid[];
extern int execstatus;
extern int errcode;

#ifdef DISCARD
# define DISCARD (void)
#endif

#ifdef sword
# define sword int
#endif

#define VER7      2

#define NA      -1  /* ANSI SQL NULL */
#define NLT      1  /* length for string null terminator */
#define DEADLOCK  60 /* ORA-00060: deadlock */
#define NO_DATA_FOUND 1403 /* ORA-01403: no data found */
#define NOT_SERIALIZABLE 8177 /* ORA-08177: transaction not
serializable */
#define SNAPSHOT_TOO_OLD 1555 /* ORA-01555: snapshot too
old */

#ifdef NULLP
# define NULLP (void *)NULL
#endif /* NULLP */

#define ADR(object) ((ub1 *)&(object))
#define SIZ(object) ((sword)sizeof(object))

typedef char date[24+NLT];
typedef char varchar2;

#define OCIERROR(errp,function)\
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(void
*)(errp),(function));

```

```

#define OCIBND(stmp, bndp, errp, sqlvar, progvl, progvl, ftype)\
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp), &(bndp), (errp), \
(text *)(sqlvar), strlen((sqlvar)), \
(progvl), (progvl), (ftype),0,0,0,OCI_DEFAULT));

#define
OCIBNDRA(stmp,bndp,errp,sqlvar,progvl,progvl,ftype,indp,alen,arco
de) \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp),&(bndp),(errp),(text
*)(sqlvar),strlen((sqlvar)), \

(progvl),(progvl),(ftype),(indp),(alen),(arcde),0,0,OCI_DEFAULT));
#define
OCIBNDRAD(stmp,bndp,errp,sqlvar,progvl,ftype,indp,ctxp,cbf_noda
ta,cbf_data) \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp),&(bndp),(errp),(text
*)(sqlvar), \
strlen((sqlvar)),0,(progvl),(ftype), \
indp,0,0,0,OCI_DATA_AT_EXEC)); \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \

OCIBindDynamic((bndp),(errp),(ctxp),(cbf_nodata),(ctxp),(cbf_data))
);

#define
OCIBNDR(stmp,bndp,errp,sqlvar,progvl,progvl,ftype,indp,alen,arcod
e) \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),(dvoid*)&(bndp),OCI_HTYPE_BIND,0,(dvoi
d**0)); \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp),&(bndp),(errp),(text
*)(sqlvar),strlen((sqlvar)), \

(progvl),(progvl),(ftype),(indp),(alen),(arcde),0,0,OCI_DEFAULT));

#define
OCIBNDRAA(stmp,bndp,errp,sqlvar,progvl,progvl,ftype,indp,alen,arc
ode,ms,cu) \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \

OCIHandleAlloc((stmp),&(bndp),OCI_HTYPE_BIND,0,(dvoid**0)); \
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp), \
OCIBindByName((stmp),&(bndp),(errp),(text
*)(sqlvar),strlen((sqlvar)), \

(progvl),(progvl),(ftype),(indp),(alen),(arcde),(ms),(cu),OCI_DEFAU
LT));

#define OCIDEFINE(stmp,dfnp,errp,pos,progvl,progvl,ftype)\

OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progvl),(progvl),(ftype), \
0,0,0,OCI_DEFAULT);

```

```

#define OCIDEF(stmp,dfnp,errp,pos,progv,proglv,ftype) \

OCIHandleAlloc((stmp),(dvoid*)&(dfnp),OCI_HTYPE_DEFINE,0,\
(dvoid**)0);\
OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),(proglv),\
(ftype),NULL,NULL,NULL,OCI_DEFAULT);\

#define
OCIDFNRA(stmp,dfnp,errp,pos,progv,proglv,ftype,indp,alen,arcodes)
\

OCIHandleAlloc((stmp),(dvoid*)&(dfnp),OCI_HTYPE_DEFINE,0,\
(dvoid**)0);\
OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
(proglv),(ftype),(indp),(alen),\
(arcodes),OCI_DEFAULT);\
/*
OCIDefineArrayOfStruct((dfnp),(errp),(proglv),\
sizeof((indp)[0]),\
sizeof((alen)[0]),\
sizeof((arcodes)[0]));
*/
/*
#define
OCIDFNRA(stmp,dfnp,errp,pos,progv,proglv,ftype,indp,alen,arcodes)
\
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp),\
OCIHandleAlloc(tpcenv,&(dfnp),OCI_HTYPE_DEFINE,0,\
(dvoid**)0);\
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp),\
OCIDefineByPos((stmp),&(dfnp),(errp),(pos),(progv),\
(proglv),(ftype),(indp),(alen),\
(arcodes),OCI_DEFAULT));\
ocierror(FILE *server_logtrans,__FILE__,__LINE__,(errp),\
OCIDefineArrayOfStruct((dfnp),(errp),(proglv),\
sizeof((indp)[0]),\
sizeof((alen)[0]),\
sizeof((arcodes)[0])));
*/

#define OBNDRV(lda,cursor,sqlvar,progv,proglv,ftype)\
if
(obndrv((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(proglv),(ftype),NA,\
(sb2 *)0, (text *)0, NA, NA))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0

#define
OBNDRA(lda,cursor,sqlvar,progv,proglv,ftype,indp,alen,arcodes)\
if
(obndra((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(proglv),(ftype),NA,\
(indp),(alen),(arcodes),(ub4*)0,(ub4*)0,(text*)0,NA,NA))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0

#define
OBNDRAA(lda,cursor,sqlvar,progv,proglv,ftype,indp,alen,arcodes,ms,cs)\
if
(obndra((cursor),(text*)(sqlvar),NA,(ub1*)(progv),(proglv),(ftype),NA,\
(indp),(alen),(arcodes),(ub4*)(ms),(ub4*)(cs),(text*)0,NA,NA))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0

#define
ODEFIN(lda,cursor,pos,buf,bufi,ftype,scale,indp,fmt,fmtl,fmtt,rten,rcode)
\
if (odefin((cursor),(pos),(ub1*)(buf),(bufi),(ftype),(scale),(indp),\
(text*)(fmt),(fmtl),(fmmt),(rten),(rcode)))

```

```

{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0

#define OEXFET(lda,cursor,nrows, cancel, exact)\
if (oexfet((cursor),(nrows),(cancel),(exact)))\
{if ((cursor)->rc == 1403) \
{i=errrpt(lda,cursor); orol(lda); return(-1);} \
else if (errrpt(lda,cursor)==RECOVER) \
{orol(lda);return(RECOVER);} \
else{orol(lda);return(-1);}}\
else\
DISCARD 0

#define OOPEN(lda,cursor)\
if (oopen((cursor),(lda),(text*)0,NA,NA,(text*)0,NA))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0

#define OPARSE(lda,cursor,sqlstm,sqll,defflg,lngflg)\
if (oparse((cursor),(sqlstm),(sb4)(sqll),(defflg),(ub4)(lngflg)))\
{errrpt(lda,cursor);return(-1);}\
else\
DISCARD 0

#define OFEN(lda,cursor,nrows)\
if (ofen((cursor),(nrows)))\
{if (errrpt(lda,cursor)==RECOVER) \
{orol(lda);return(RECOVER);} \
else{orol(lda);return(-1);}}\
else\
DISCARD 0

#define OEXEC(lda,cursor)\
if (oexec((cursor)))\
{if (errrpt(lda,cursor)==RECOVER) \
{orol(lda);return(RECOVER);} \
else{orol(lda);return(-1);}}\
else\
DISCARD 0

#define OCOM(lda,cursor)\
if (ocom((lda))) \
{errrpt(lda,cursor);orol(lda);return(-1);}\
else\
DISCARD 0

#define OEXN(lda,cursor,itors,rowoff)\
if (oexn((cursor),(itors),(rowoff)) \
{if (errrpt(lda,cursor)==RECOVER) \
{orol(lda);return(RECOVER);} \
else{orol(lda);return(-1);}}\
else\
DISCARD 0

#endif

```

A.3 TM Code

Com file set 0,1 and 50 of Com files, 0 through 50.

tpccCom0/dlldata.c

/******

DLData file -- generated by MIDL compiler

DO NOT ALTER THIS FILE

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

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

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

```
#include <rpcproxy.h>
```

```
#ifdef __cplusplus
extern "C" {
#endif
```

```
EXTERN_PROXY_FILE( tpccCom0 )
```

```
PROXYFILE_LIST_START
/* Start of list */
REFERENCE_PROXY_FILE( tpccCom0 ),
/* End of list */
PROXYFILE_LIST_END
```

```
DLLDATA_ROUTINES( aProxyFileList, GET_DLL_CLSID )
```

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

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

tpccCom0/dlldata.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 "tpccCom0_p.c"
```

```
#endif // _MERGE_PROXYSTUB
```

tpccCom0/dlldata.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
```

tpccCom0/resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by tpccCom0.rc
//
#define IDS_PROJNAME 100
#define IDR_TPCCCOM0 101
#define IDR_TPCC_COM0 102

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifdef 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
```

tpccCom0/stdafx.cpp

```
// stdafx.cpp : source file that includes just the standard includes
// tpccCom0.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information
```

```
#include "stdafx.h"
```

tpccCom0/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;

tpccCom0/tpcc_com0.cpp

// tpcc_com0.cpp : Implementation of Ctpcc_com0

#include "stdafx.h"
#include "tpcc_com0.h"

#include "..\tpccIsapi\tpcc.h"

#include "..\tpcc_com0.h"
#ifdef DB2
#include <db2tpcc.h>
#endif
#ifdef ORA
#include <ora_tpcc.h>
#endif

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

char tmpbuff[128];

// Ctpcc_com0
HRESULT Ctpcc_com0::Activate()
{
    HRESULT hr = GetObjectContext(&m_spObjectContext);
    if (SUCCEEDED(hr))
    {
        DEBUGMSG("Object assigned to thread."<<endl);
        return S_OK;
    }
    return hr;
}

BOOL Ctpcc_com0::CanBePooled()
{
    DEBUGMSG("CanBePooled() returning true."<<endl);
    return TRUE;
}

void Ctpcc_com0::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_com0::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_com0::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);

    do_stok(stok,connectHandle);

```

```

    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_com0::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_com0::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_com0::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);

    do_ords(ords,connectHandle);

    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_com0::doDBInfo(void)
{
    //DUMMY FUNCTION TO ALLOW ISAPI TO CALL COM
    FUNCTION DURING LOGIN SO CONNECTIONS
    //TO DATABASE WILL BE SETUP.....I HOPE.
    DEBUGMSG("Stub function to warm object pool"<<endl);

    return S_OK;
}

/*
*****
** Name      : loadLibrary
** Description :
**      Function loads appropiate db library based on
**      registry setting
** Parameters:
** Returns   :
**      int - return code
** Comments  :
**
*****
*/

Ctpcc_com0::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
{
    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,"disconnect_db")
) == NULL )
        return ERR_DISCONNECT_ADDRESS_NOT_FOUND;
    DEBUGMSG("do_disconnect
address:"<<DEBUGADDRESS(do_disconnect)<<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
address:"<<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 registry
**      dbName, dbUserName, and dbUserPassword
*****
*/

Ctpcc_com0::readRegistry()
{
    //open registry key
    HKEY registryKey;
    DWORD regType;
    char value[MAX_STRING_LEN];
    DWORD regValue;
    DWORD regValueSize = MAX_STRING_LEN;

    DEBUGMSG("Entered readRegistry(), opening key:"<<
REGISTRY_SUB_KEY <<endl);
    //open up registry key
    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTRY_SUB_
KEY,0,KEY_READ,&registryKey) == ERROR_SUCCESS)
    {
        DEBUGMSG(REGISTRY_SUB_KEY<<" open, getting
database type from key"<<endl);
        regValueSize = sizeof(value);

```

```

        if
        (RegQueryValueEx(registryKey,DB_TYPE,0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
            strcpy(dbType,value);
        DEBUGMSG("Database type:"<<dbType<<" from registry
key."<<endl);

        DEBUGMSG("Getting database name from registry
key."<<endl);
        regValueSize = sizeof(value);
        if
        (RegQueryValueEx(registryKey,DB_NAME,0,&regType,(BYTE *)
&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(registryKey,NULL_DB,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
            nullDB = regValue;
        DEBUGMSG("Null database flag:"<<nullDB<<endl);

        return OK;
    }

    DEBUGMSG("Error, unable to open registry key."<<endl);
    return ERR_UNABLE_TO_OPEN_REG;
}

/*
*****
** Name      : connectDB
** Description :
**      Function connects to the db
** Parameters:
** Returns   :
**      int - return code
** Comments  :
*****
*/
Ctpcc_com0::connectDB()
{
    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)
        {
            DEBUGMSG("Setting Context handle in use to true"<<endl);
            connectHandleInUse = 1;
            connected = do_connection(dbName,&connectHandle);
            if(connected != OK)
            {
                DEBUGMSG("Object do_connect failed,
rc:"<<connected<<endl);
                ERRORMSG("Object do_connect failed,
rc:"<<connected<<endl);
                return connected;
            }
            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;
}

```

tpccCom0/tpccCom0.cpp

```

// tpccCom0.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 "tpccCom0.h"
#include "dlldatax.h"

class CtpccCom0Module : public CAtlDllModuleT<
CtpccCom0Module >
{
public :
    DECLARE_LIBID(LIBID_tpccCom0Lib)
    DECLARE_REGISTRY_APPID_RESOURCEID(IDR_TPCCCOM0
, "{55A71599-F6F1-4AD9-BD70-E1B3178D5E9A}")
};

CtpccCom0Module _AtlModule;

// 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;
    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;
}

```

tpccCom0/tpccCom0.def

; tpccCom0.def : Declares the module parameters.

```

LIBRARY "tpccCom0.DLL"

EXPORTS
    DllCanUnloadNow PRIVATE
    DllGetClassObject PRIVATE
    DllRegisterServer PRIVATE
    DllUnregisterServer PRIVATE

```

tpccCom0/tpcc_com0.h

// tpcc_com0.h : Declaration of the Ctpcc_com0

```

#pragma once
#include "tpccCom0.h"
#include "resource.h"// main symbols
#include <comsvcs.h>

```

```

#include "..\tpcc\api\tpcc.h"
#ifdef ORACLE
#define ORA_NT
#include <stdio.h>
#include <time.h>
#include <ora_tpcc.h>
#include <plora.h>
#endif
#define NULL_DB "nullIDB"

```

```

static HINSTANCE dbInstance = NULL;

static CRITICAL_SECTION debugMutex;
static CRITICAL_SECTION errorMutex;

static int comServerID = 0;
static ofstream debugStream;
static ofstream errorStream;

```

```

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);

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_com0
class ATL_NO_VTABLE Ctpcc_com0 :
public CComObjectRootEx<CComMultiThreadModel>,
public IObjectControl,
public CComCoClass<Ctpcc_com0, &CLSID_tpcc_com0>,
public Itpcc_com0
{
public:
    Ctpcc_com0()
    {
        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_debug.txt
");
                debugStream.rdbuf( )->open(comLogFile,ios_base::in |
ios_base::out | ios_base::app);

                debugFileOpen = 1;
            }
        }

        //open error log file
        if(!errorFileOpen)
        {
            InitializeCriticalSection(&errorMutex);

            char errorLogFile[128];

            sprintf(errorLogFile,"C:\\inetpub\\wwwroot\\tpcc\\comLog_err.txt");

            errorStream.rdbuf( )->open(errorLogFile,ios_base::in |
ios_base::out | ios_base::app);

            errorFileOpen=1;
        }

        //get registry values

```

```

        if((rc = readRegistry()) != OK)
        {
            ERRORMSG("Unable to open registry key " <<
REGISTRY_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_COM0)

    BEGIN_COM_MAP(Ctpcc_com0)
        COM_INTERFACE_ENTRY(Itpcc_com0)
        COM_INTERFACE_ENTRY(IObjectControl)
    END_COM_MAP()

    // IObjectControl
    public:
        STDMETHOD(Activate)();
        STDMETHOD_(BOOL, CanBePooled)();
        STDMETHOD_(void, Deactivate)();
        CComPtr<IObjectContext> m_spObjectContext;

    // Itpcc_com0
    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:
    //db2 specific context
    void *connectHandle;
    int loadLibrary();
    int readRegistry();
    int connectDB();

};

OBJECT_ENTRY_AUTO(__uuidof(tpcc_com0), Ctpcc_com0)

tpccCom0/tpccCom0.h

/* this ALWAYS GENERATED file contains the definitions for the
interfaces */

/* File created by MIDL compiler version 6.00.0361 */
/* at Tue Jun 08 12:21:49 2004
*/
/* Compiler settings for .tpccCom0.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 tpccCom0_h__
#define tpccCom0_h__

#ifdef _MSC_VER && (_MSC_VER >= 1020)
#pragma once
#endif

/* Forward Declarations */

#ifndef __Itpcc_com0_FWD_DEFINED__
#define __Itpcc_com0_FWD_DEFINED__
typedef interface Itpcc_com0 Itpcc_com0;
#endif /* __Itpcc_com0_FWD_DEFINED__ */

#ifndef __tpcc_com0_FWD_DEFINED__
#define __tpcc_com0_FWD_DEFINED__

```

```

#ifdef __cplusplus
typedef class tpcc_com0 tpcc_com0;
#else
typedef struct tpcc_com0 tpcc_com0;
#endif /* __cplusplus */

#endif /* __tpcc_com0_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 __Itpcc_com0_INTERFACE_DEFINED__
#define __Itpcc_com0_INTERFACE_DEFINED__

/* interface Itpcc_com0 */
/* [unique][helpstring][uuid][object] */

EXTERN_C const IID IID_Itpcc_com0;

#if defined(__cplusplus) && !defined(CINTERFACE)

    MIDL_INTERFACE("98834483-D35E-4CB2-87F9-85BABD2D914E")
    Itpcc_com0 : 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_com0Vtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE *QueryInterface )(


```

```

        Itpcc_com0 * This,
        /* [in] */ REFIID riid,
        /* [iid_is][out] */ void **ppvObject);

ULONG ( STDMETHODCALLTYPE *AddRef )(
    Itpcc_com0 * This);

ULONG ( STDMETHODCALLTYPE *Release )(
    Itpcc_com0 * This);

/* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doStockLevel )(
    Itpcc_com0 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

/* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doNewOrder )(
    Itpcc_com0 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

/* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doPayment )(
    Itpcc_com0 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

/* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doOrderStatus )(
    Itpcc_com0 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

/* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doDBInfo )(
    Itpcc_com0 * This);

/* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doSetComplete )(
    Itpcc_com0 * This);

    END_INTERFACE
} Itpcc_com0Vtbl;

interface Itpcc_com0
{
    CONST_VTBL struct Itpcc_com0Vtbl *lpVtbl;
};

#ifdef COBJMACROS

#define Itpcc_com0_QueryInterface(This,riid,ppvObject) \
    (This)->lpVtbl->QueryInterface(This,riid,ppvObject)

#define Itpcc_com0_AddRef(This) \
    (This)->lpVtbl->AddRef(This)

#define Itpcc_com0_Release(This) \
    (This)->lpVtbl->Release(This)

#define Itpcc_com0_doStockLevel(This,size,buffer) \
    (This)->lpVtbl->doStockLevel(This,size,buffer)

#define Itpcc_com0_doNewOrder(This,size,buffer) \
    (This)->lpVtbl->doNewOrder(This,size,buffer)

#define Itpcc_com0_doPayment(This,size,buffer) \

```

```

    (This)->lpVtbl->doPayment(This,size,buffer)

#define Itpcc_com0_doOrderStatus(This,size,buffer) \
    (This)->lpVtbl->doOrderStatus(This,size,buffer)

#define Itpcc_com0_doDBInfo(This) \
    (This)->lpVtbl->doDBInfo(This)

#define Itpcc_com0_doSetComplete(This) \
    (This)->lpVtbl->doSetComplete(This)

#endif /* COBJMACROS */

/* C style interface */

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com0_doStockLevel_Proxy(
    Itpcc_com0 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com0_doStockLevel_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com0_doNewOrder_Proxy(
    Itpcc_com0 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com0_doNewOrder_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com0_doPayment_Proxy(
    Itpcc_com0 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com0_doPayment_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com0_doOrderStatus_Proxy(
    Itpcc_com0 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com0_doOrderStatus_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

```

```

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com0_doDBInfo_Proxy(
    Itpcc_com0 * This);

void __RPC_STUB Itpcc_com0_doDBInfo_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com0_doSetComplete_Proxy(
    Itpcc_com0 * This);

void __RPC_STUB Itpcc_com0_doSetComplete_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

#endif /* __Itpcc_com0_INTERFACE_DEFINED__ */

#ifndef __tpccCom0Lib_LIBRARY_DEFINED__
#define __tpccCom0Lib_LIBRARY_DEFINED__

/* library tpccCom0Lib */
/* [helpstring][version][uuid] */

EXTERN_C const IID LIBID_tpccCom0Lib;

EXTERN_C const CLSID CLSID_tpcc_com0;

#ifdef __cplusplus

class DECLSPEC_UUID("FB56BC3B-269F-4C81-907F-08EC0936EADF")
tpcc_com0;
#endif
#endif /* __tpccCom0Lib_LIBRARY_DEFINED__ */

/* Additional Prototypes for ALL interfaces */

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif

```

tpccCom0/tpccCom0_i.c

```

/* this ALWAYS GENERATED file contains the IIDs and CLSIDs */

/* link this file in with the server and any clients */

/* File created by MIDL compiler version 6.00.0361 */

```

```

/* at Tue Jun 08 12:21:49 2004
*/
/* Compiler settings for .ItpccCom0.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( )

#ifdef !defined(_M_IA64) && !defined(_M_AMD64)

#pragma warning( disable: 4049 ) /* more than 64k source lines */

#ifndef __cplusplus
extern "C"{
#endif

#include <rpc.h>
#include <rpcndr.h>

#ifndef _MIDL_USE_GUIDDEF_

#ifndef INITGUID
#define INITGUID
#include <guiddef.h>
#undef INITGUID
#else
#include <guiddef.h>
#endif
#endif

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8) \
    DEFINE_GUID(name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8)

#else /* !_MIDL_USE_GUIDDEF_ */

#ifndef __IID_DEFINED__
#define __IID_DEFINED__

typedef struct _IID
{
    unsigned long x;
    unsigned short s1;
    unsigned short s2;
    unsigned char c[8];
} IID;

#endif /* __IID_DEFINED__ */

#ifndef CLSID_DEFINED
#define CLSID_DEFINED
typedef IID CLSID;
#endif /* CLSID_DEFINED */

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8) \
    const type name = {l,w1,w2,{b1,b2,b3,b4,b5,b6,b7,b8}}

#endif /* !_MIDL_USE_GUIDDEF_ */

MIDL_DEFINE_GUID(IID,
IID_Itpcc_com0,0x98834483,0xD35E,0x4CB2,0x87,0xF9,0x85,0xBA,0xBD,0x2D,0x91,0x4E);

```

```
MIDL_DEFINE_GUID(IID,
LIBID_tpccCom0Lib,0x7280DF7D,0xDFEF,0x445F,0x89,0xA4,0x04,
0x98,0x82,0x78,0xF8,0xEE);
```

```
MIDL_DEFINE_GUID(CLSID,
CLSID_tpcc_com0,0xFB56BC3B,0x269F,0x4C81,0x90,0x7F,0x08,0
xEC,0x09,0x36,0xEA,0xDF);
```

```
#undef MIDL_DEFINE_GUID
```

```
#ifdef __cplusplus
}
#endif
```

```
#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/
```

tpccCom0/tpccCom0.idl

```
// tpccCom0.idl : IDL source for tpccCom0
//
```

```
// This file will be processed by the MIDL tool to
// produce the type library (tpccCom0.tlb) and marshalling code.
```

```
import "oidl.idl";
import "ocidl.idl";
```

```
[
    object,
    uuid(98834483-D35E-4CB2-87F9-85BABD2D914E),
    // oleautomation,
    // nonextensible,
    helpstring("Itpcc_com0 Interface"),
    pointer_default(unique)
]
interface Itpcc_com0 : 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(7280DF7D-DFEF-445F-89A4-04988278F8EE),
    version(1.0),
    helpstring("tpccCom0 1.0 Type Library")
]
library tpccCom0Lib
{
    importlib("stdole2.tlb");
    [
        uuid(FB56BC3B-269F-4C81-907F-08EC0936EADF),
        helpstring("tpcc_com0 Class")
    ]
    coclass tpcc_com0
    {
        [default] interface Itpcc_com0;
    };
};
```

tpccCom0/tpccCom0.p.c

```
/* this ALWAYS GENERATED file contains the proxy stub code */
```

```
/* File created by MIDL compiler version 6.00.0361 */
```

```
/* at Tue Jun 08 12:21:49 2004
```

```
*/
```

```
/* Compiler settings for .tpccCom0.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 "tpccCom0.h"
```

```
#define TYPE_FORMAT_STRING_SIZE 27
```

```
#define PROC_FORMAT_STRING_SIZE 229
```

```
#define TRANSMIT_AS_TABLE_SIZE 0
```

```
#define WIRE_MARSHAL_TABLE_SIZE 0
```

```
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,0x10,0
x48,0x60}}, {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 Itppcc_com0_ServerInfo;
extern const MIDL_STUBLESS_PROXY_INFO
Itppcc_com0_ProxyInfo;

#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 doStockLevel */

        0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 2 */ NdrFcLong( 0x0 ), /* 0 */
/* 6 */ NdrFcShort( 0x3 ), /* 3 */
/* 8 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 10 */ NdrFcShort( 0x1c ), /* 28 */
/* 12 */ NdrFcShort( 0x8 ), /* 8 */
/* 14 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */
/* 16 */ 0x8, /* 8 */
        0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 18 */ NdrFcShort( 0x1 ), /* 1 */
/* 20 */ NdrFcShort( 0x1 ), /* 1 */
/* 22 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 24 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 26 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 28 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Parameter buffer */

/* 30 */ NdrFcShort( 0x201b ), /* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 32 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 34 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

```

```

/* 36 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 38 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 40 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Procedure doNewOrder */

/* 42 */ 0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 44 */ NdrFcLong( 0x0 ), /* 0 */
/* 48 */ NdrFcShort( 0x4 ), /* 4 */
/* 50 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 52 */ NdrFcShort( 0x1c ), /* 28 */
/* 54 */ NdrFcShort( 0x8 ), /* 8 */
/* 56 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */
/* 58 */ 0x8, /* 8 */
        0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 60 */ NdrFcShort( 0x1 ), /* 1 */
/* 62 */ NdrFcShort( 0x1 ), /* 1 */
/* 64 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 66 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 68 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 70 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Parameter buffer */

/* 72 */ NdrFcShort( 0x201b ), /* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 74 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 76 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 78 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 80 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 82 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Procedure doPayment */

/* 84 */ 0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 86 */ NdrFcLong( 0x0 ), /* 0 */
/* 90 */ NdrFcShort( 0x5 ), /* 5 */
/* 92 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 94 */ NdrFcShort( 0x1c ), /* 28 */
/* 96 */ NdrFcShort( 0x8 ), /* 8 */
/* 98 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */
/* 100 */ 0x8, /* 8 */
        0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 102 */ NdrFcShort( 0x1 ), /* 1 */
/* 104 */ NdrFcShort( 0x1 ), /* 1 */
/* 106 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 108 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 110 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 112 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Parameter buffer */

```



```

/* 114 */NdrFcShort( 0x201b ),/* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 116 */NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 118 */NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 120 */NdrFcShort( 0x70 ),/* Flags: out, return, base type, */
/* 122 */NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 124 */0x8, /* FC_LONG */
0x0, /* 0 */

/* Procedure doOrderStatus */

/* 126 */0x33, /* FC_AUTO_HANDLE */
0x6c, /* Old Flags: object, Oi2 */
/* 128 */NdrFcLong( 0x0 ),/* 0 */
/* 132 */NdrFcShort( 0x6 ), /* 6 */
/* 134 */NdrFcShort( 0x10 ),/* x86 Stack size/offset = 16 */
/* 136 */NdrFcShort( 0x1c ),/* 28 */
/* 138 */NdrFcShort( 0x8 ), /* 8 */
/* 140 */0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
0x3, /* 3 */
/* 142 */0x8, /* 8 */
0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 144 */NdrFcShort( 0x1 ), /* 1 */
/* 146 */NdrFcShort( 0x1 ), /* 1 */
/* 148 */NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 150 */NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 152 */NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 154 */0x8, /* FC_LONG */
0x0, /* 0 */

/* Parameter buffer */

/* 156 */NdrFcShort( 0x201b ),/* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 158 */NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 160 */NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 162 */NdrFcShort( 0x70 ),/* Flags: out, return, base type, */
/* 164 */NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 166 */0x8, /* FC_LONG */
0x0, /* 0 */

/* Procedure doDBInfo */

/* 168 */0x33, /* FC_AUTO_HANDLE */
0x6c, /* Old Flags: object, Oi2 */
/* 170 */NdrFcLong( 0x0 ),/* 0 */
/* 174 */NdrFcShort( 0x7 ), /* 7 */
/* 176 */NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 178 */NdrFcShort( 0x0 ), /* 0 */
/* 180 */NdrFcShort( 0x8 ), /* 8 */
/* 182 */0x44, /* Oi2 Flags: has return, has ext, */
0x1, /* 1 */
/* 184 */0x8, /* 8 */
0x1, /* Ext Flags: new corr desc, */
/* 186 */NdrFcShort( 0x0 ), /* 0 */
/* 188 */NdrFcShort( 0x0 ), /* 0 */
/* 190 */NdrFcShort( 0x0 ), /* 0 */

/* Return value */

```

```

/* 192 */NdrFcShort( 0x70 ),/* Flags: out, return, base type, */
/* 194 */NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 196 */0x8, /* FC_LONG */
0x0, /* 0 */

/* Procedure doSetComplete */

/* 198 */0x33, /* FC_AUTO_HANDLE */
0x6c, /* Old Flags: object, Oi2 */
/* 200 */NdrFcLong( 0x0 ),/* 0 */
/* 204 */NdrFcShort( 0x8 ), /* 8 */
/* 206 */NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 208 */NdrFcShort( 0x0 ), /* 0 */
/* 210 */NdrFcShort( 0x8 ), /* 8 */
/* 212 */0x44, /* Oi2 Flags: has return, has ext, */
0x1, /* 1 */
/* 214 */0x8, /* 8 */
0x1, /* Ext Flags: new corr desc, */
/* 216 */NdrFcShort( 0x0 ), /* 0 */
/* 218 */NdrFcShort( 0x0 ), /* 0 */
/* 220 */NdrFcShort( 0x0 ), /* 0 */

/* Return value */

/* 222 */NdrFcShort( 0x70 ),/* Flags: out, return, base type, */
/* 224 */NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 226 */0x8, /* FC_LONG */
0x0, /* 0 */

0x0
}
};

static const MIDL_TYPE_FORMAT_STRING
_MIDL_TypeFormatString =
{
0,
{
NdrFcShort( 0x0 ), /* 0 */
/* 2 */
0x11, 0x8, /* FC_RP [simple_pointer] */
/* 4 */ 0x8, /* FC_LONG */
0x5c, /* FC_PAD */
/* 6 */
0x11, 0x14, /* FC_RP [allocated_on_stack] [pointer_deref] */
/* 8 */ NdrFcShort( 0x2 ), /* Offset= 2 (10) */
/* 10 */
0x13, 0x0, /* FC_OP */
/* 12 */ NdrFcShort( 0x2 ), /* Offset= 2 (14) */
/* 14 */
0x1b, /* FC_CARRAY */
0x0, /* 0 */
/* 16 */ NdrFcShort( 0x1 ), /* 1 */
/* 18 */ 0x28, /* Corr desc: parameter, FC_LONG */
0x54, /* FC_DEREFERENCE */
/* 20 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 22 */ NdrFcShort( 0x1 ), /* Corr flags: early, */
/* 24 */ 0x2, /* FC_CHAR */
0x5b, /* FC_END */

0x0
}
};

/* Object interface: IUnknown, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x
00,0x00,0x46}} */

/* Object interface: Itpcc_com0, ver. 0.0,

```

```
GUID={0x98834483,0xD35E,0x4CB2,{0x87,0xF9,0x85,0xBA,0xBD,
0x2D,0x91,0x4E}} */
```

```
#pragma code_seg(".orpc")
static const unsigned short Itpcc_com0_FormatStringOffsetTable[] =
{
    0,
    42,
    84,
    126,
    168,
    198
};
```

```
static const MIDL_STUBLESS_PROXY_INFO
Itpcc_com0_ProxyInfo =
{
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &Itpcc_com0_FormatStringOffsetTable[-3],
    0,
    0,
    0
};
```

```
static const MIDL_SERVER_INFO Itpcc_com0_ServerInfo =
{
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &Itpcc_com0_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0
};
```

```
CINTERFACE_PROXY_VTABLE(9) _Itpcc_com0ProxyVtbl =
{
    &Itpcc_com0_ProxyInfo,
    &IID_Itpcc_com0,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    (void *) (INT_PTR) -1 /* Itpcc_com0::doStockLevel */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com0::doNewOrder */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com0::doPayment */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com0::doOrderStatus */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com0::doDBInfo */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com0::doSetComplete */
};
```

```
const CInterfaceStubVtbl _Itpcc_com0StubVtbl =
{
    &IID_Itpcc_com0,
    &Itpcc_com0_ServerInfo,
    9,
    0, /* pure interpreted */
    CStdStubBuffer_METHODS
};
```

```
static const MIDL_STUB_DESC Object_StubDesc =
{
    0,
    NdrOleAllocate,
    NdrOleFree,
    0,
    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,
0,
0, /* notify & notify_flag routine table */
0x1, /* MIDL flag */
0, /* cs routines */
0, /* proxy/server info */
0 /* Reserved5 */
};
```

```
const CInterfaceProxyVtbl * _tpccCom0_ProxyVtblList[] =
{
    (CInterfaceProxyVtbl *) &_Itpcc_com0ProxyVtbl,
    0
};
```

```
const CInterfaceStubVtbl * _tpccCom0_StubVtblList[] =
{
    (CInterfaceStubVtbl *) &_Itpcc_com0StubVtbl,
    0
};
```

```
PCInterfaceName const _tpccCom0_InterfaceNamesList[] =
{
    "Itpcc_com0",
    0
};
```

```
#define _tpccCom0_CHECK_IID(n)
IID_GENERIC_CHECK_IID( _tpccCom0, pIID, n)
```

```
int __stdcall _tpccCom0_IID_Lookup( const IID * pIID, int * pIndex )
{
    if(! _tpccCom0_CHECK_IID(0))
    {
        *pIndex = 0;
        return 1;
    }

    return 0;
}
```

```
const ExtendedProxyFileInfo tpccCom0_ProxyFileInfo =
{
    (PCInterfaceProxyVtblList *) &_tpccCom0_ProxyVtblList,
    (PCInterfaceStubVtblList *) &_tpccCom0_StubVtblList,
    (const PCInterfaceName *) &_tpccCom0_InterfaceNamesList,
    0, // no delegation
    &_tpccCom0_IID_Lookup,
    1,
    2,
    0, /* table of [async_uuid] interfaces */
    0, /* Filler1 */
    0, /* Filler2 */
    0 /* Filler3 */
};
#ifdef _MSC_VER >= 1200
#pragma warning(pop)
#endif
```

```
#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/
```

tpccCom0/tpccCom0.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)
#ifndef _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 \"\"tpccCom0.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", "tpccCom0.dll"
            VALUE "OriginalFilename", "tpccCom0.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_TPCCCOM0      REGISTRY      "tpccCom0.rgs"
IDR_TPCC_COM0     REGISTRY      "tpcc_com0.rgs"

////////////////////////////////////
//
// String Table
//

STRINGTABLE
BEGIN
    IDS_PROJNAME      "tpccCom0"
END

#endif // English (U.S.) resources
////////////////////////////////////

#ifndef APSTUDIO_INVOKED
////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
1 TYPELIB "tpccCom0.tlb"

////////////////////////////////////
#endif // not APSTUDIO_INVOKED

```

tpccCom0/tpcc_com0.rgs

```
HKCR
{
    tpccCom0.tpcc_com0.1 = s 'tpcc_com0 Class'
    {
        CLSID = s '{FB56BC3B-269F-4C81-907F-08EC0936EADF}'
    }
    tpccCom0.tpcc_com0 = s 'tpcc_com0 Class'
    {
        CLSID = s '{FB56BC3B-269F-4C81-907F-08EC0936EADF}'
        CurVer = s 'tpccCom0.tpcc_com0.1'
    }
    NoRemove CLSID
    {
        ForceRemove {FB56BC3B-269F-4C81-907F-08EC0936EADF}
    } = s 'tpcc_com0 Class'
    {
        ProgID = s 'tpccCom0.tpcc_com0.1'
        VersionIndependentProgID = s 'tpccCom0.tpcc_com0'
        InprocServer32 = s '%MODULE%'
    }
}

```

```

        val ThreadingModel = s 'Both'
    }
    val AppID = s '%APPID%'
    'TypeLib' = s '{7280DF7D-DFEF-445F-89A4-04988278F8EE}'
}
}
}

```

tpccCom0/tpccCom0.rgs

```

HKCR
{
    NoRemove AppID
    {
        '%APPID%' = s 'tpccCom0'
        'tpccCom0.DLL'
        {
            val AppID = s '%APPID%'
        }
    }
}

```

tpccCom1/dlldata.c

/******

DllData file -- generated by MIDL compiler

DO NOT ALTER THIS FILE

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

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

*****/

```
#include <rpcproxy.h>
```

```

#ifdef __cplusplus
extern "C" {
#endif

```

```
EXTERN_PROXY_FILE( tpccCom1 )
```

```

PROXYFILE_LIST_START
/* Start of list */
    REFERENCE_PROXY_FILE( tpccCom1 ),
/* End of list */
PROXYFILE_LIST_END

```

```
DLLDATA_ROUTINES( aProxyFileList, GET_DLL_CLSID )
```

```

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

```

/* end of generated dlldata file */

tpccCom1/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 "tpccCom1_p.c"

```

```
#endif // _MERGE_PROXYSTUB
```

tpccCom1/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
```

tpccCom1/resource.h

```

//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by tpccCom1.rc
//
#define IDS_PROJNAME            100
#define IDR_TPCCCOM1            101
#define IDR_TPCC_COM1           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

```

tpccCom1/stdafx.cpp

// stdafx.cpp : source file that includes just the standard includes
// tpccCom1.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

```
#include "stdafx.h"
```

tpccCom1/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_NAMESPACES

#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;

tpccCom1/tpcc_com1.cpp

// tpcc_com1.cpp : Implementation of Ctpcc_com1

#include "stdafx.h"
#include "tpcc_com1.h"

#include "..\tpcc\api\tpcc.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_com1
HRESULT Ctpcc_com1::Activate()
{
    HRESULT hr = GetObjectContext(&m_spObjectContext);
    if (SUCCEEDED(hr))
    {
        DEBUGMSG("Object assigned to thread."<<endl);
        return S_OK;
    }
    return hr;
}

BOOL Ctpcc_com1::CanBePooled()
{
    DEBUGMSG("CanBePooled() returning true"<<endl);
    return TRUE;
}

void Ctpcc_com1::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_com1::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_com1::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);

    do_stok(stok,connectHandle);

    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_com1::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_com1::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_com1::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);

    do_ords(ords,connectHandle);

    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_com1::doDBInfo(void)
{
    //DUMMY FUNCTION TO ALLOW ISAPI TO CALL COM
    FUNCTION DURING LOGIN SO CONNECTIONS
    //TO DATABASE WILL BE SETUP.....I HOPE.

```

```

    DEBUGMSG("Stub function to warm object pool"<<endl);

    return S_OK;
}

/*
*****
** Name      : loadLibrary
** Description :
**          Function loads appropiate db library based on
**          registry setting
** Parameters:
** Returns    :
**          int - return code
** Comments   :
**
*****
*/

Ctpcc_com1::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
    {
        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,"disconnect_db")
) == NULL )
        return ERR_DISCONNECT_ADDRESS_NOT_FOUND;
    DEBUGMSG("do_disconnect
address:"<<DEBUGADDRESS(do_disconnect)<<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
address:"<<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 registry
**           dbName, dbUserName, and dbUserPassword
*****
*/

Ctpcc_com1::readRegistry()
{
    //open registry key
    HKEY registryKey;
    DWORD regType;
    char value[MAX_STRING_LEN];
    DWORD regValue;
    DWORD regValueSize = MAX_STRING_LEN;

    DEBUGMSG("Entered readRegistry(), opening key:"<<
REGISTRY_SUB_KEY <<endl);
    //open up registry key
    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTRY_SUB_
KEY,0,KEY_READ,&registryKey) == ERROR_SUCCESS)
    {
        DEBUGMSG(REGISTRY_SUB_KEY<<" open, getting
database type from key"<<endl);
        regValueSize = sizeof(value);
        if
        (RegQueryValueEx(registryKey,DB_TYPE,0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
            strcpy(dbType,value);
        DEBUGMSG("Database type:"<<dbType<<" from registry
key."<<endl);

        DEBUGMSG("Getting database name from registry
key."<<endl);
        regValueSize = sizeof(value);
        if
        (RegQueryValueEx(registryKey,DB_NAME,0,&regType,(BYTE *)
&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(registryKey,NULL_DB,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
            nullDB = regValue;
        DEBUGMSG("Null database flag:"<<nullDB<<endl);

        return OK;
    }

    DEBUGMSG("Error, unable to open registry key."<<endl);
    return ERR_UNABLE_TO_OPEN_REG;
}

/*
*****
** Name      : connectDB
** Description :
**           Function connects to the db
** Parameters:
** Returns   :
**           int - return code
** Comments  :
*****
*/

Ctpcc_com1::connectDB()
{
    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)
    {
        DEBUGMSG("Setting Context handle in use to true"<<endl);
        connectHandleInUse = 1;
        connected = do_connection(dbName,&connectHandle);
        if(connected != OK)
        {
            DEBUGMSG("Object do_connect failed,
rc:"<<connected<<endl);
            ERRORMSG("Object do_connect failed,
rc:"<<connected<<endl);
            return connected;
        }
        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;
}

```

tpccCom1/tpccCom1.cpp

```

// tpccCom1.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 "tpccCom1.h"
#include "dlldatax.h"

class CtpccCom1Module : public CAtlDllModuleT<
CtpccCom1Module >
{
public :
    DECLARE_LIBID(LIBID_tpccCom1Lib)
    DECLARE_REGISTRY_APPID_RESOURCEID(IDR_TPCCCOM1
, "{8D51801C-E25F-44E5-8309-4A6116A6B795}")
};

CtpccCom1Module _AtlModule;

// 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;
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;
}

```

tpccCom1/tpccCom1.def

; tpccCom1.def : Declares the module parameters.

```

LIBRARY "tpccCom1.DLL"

EXPORTS
    DllCanUnloadNow PRIVATE
    DllGetClassObject PRIVATE
    DllRegisterServer PRIVATE
    DllUnregisterServer PRIVATE

```

tpccCom1/tpcc_com1.h

```
// tpcc_com1.h : Declaration of the Ctpcc_com1

#pragma once
#include "tpccCom1.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 int comServerID = 0;
static ofstream debugStream;
static ofstream errorStream;
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);

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_com1
class ATL_NO_VTABLE Ctpcc_com1 :
public CComObjectRootEx<CComMultiThreadModel>,
public IObjectControl,
public CComCoClass<Ctpcc_com1, &CLSID_tpcc_com1>,
public Itpcc_com1
{
public:
Ctpcc_com1()
{
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_debug.txt
");
debugStream.rdbuf( )->open(comLogFile,ios_base::in |
ios_base::out | ios_base::app);

debugFileOpen = 1;
}
}

//open error log file
if(!errorFileOpen)
{
InitializeCriticalSection(&errorMutex);

char errorLogFile[128];

sprintf(errorLogFile,"C:\\inetpub\\wwwroot\\tpcc\\comLog_err.txt");

errorStream.rdbuf( )->open(errorLogFile,ios_base::in |
ios_base::out | ios_base::app);

errorFileOpen=1;
}

//get registry values
if((rc = readRegistry()) != OK)
{
ERRORMSG("Unable to open registry key " <<
REGISTRY_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_COM1)

BEGIN_COM_MAP(Ctpcc_com1)
    COM_INTERFACE_ENTRY(Itpcc_com1)
    COM_INTERFACE_ENTRY(IObjectControl)
END_COM_MAP()

// IObjectControl
public:
    STDMETHODCALLTYPE(Activate());
    STDMETHODCALLTYPE(BOOL, CanBePooled());
    STDMETHODCALLTYPE(void, Deactivate());
    CComPtr<IObjectContext> m_spObjectContext;

// Itpcc_com1
public:
    STDMETHODCALLTYPE(doStockLevel)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doNewOrder)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doPayment)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doOrderStatus)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doDBInfo)(void);
    STDMETHODCALLTYPE(doSetComplete)(void);

    int connected;
    int connectHandleInUse;

private:
    //db2 specific context
    void *connectHandle;
    int loadLibrary();
    int readRegistry();
    int connectDB();

};

OBJECT_ENTRY_AUTO(__uuidof(tpcc_com1), Ctpcc_com1)

tpccCom1/tpccCom1.h

/* this ALWAYS GENERATED file contains the definitions for the
interfaces */

/* File created by MIDL compiler version 6.00.0361 */
/* at Fri May 07 15:35:12 2004
*/
/* Compiler settings for .\tpccCom1.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 __tpccCom1_h__
#define __tpccCom1_h__

#if defined(_MSC_VER) && (_MSC_VER >= 1020)
#pragma once
#endif

/* Forward Declarations */

#ifndef __Itpcc_com1_FWD_DEFINED__
#define __Itpcc_com1_FWD_DEFINED__
typedef interface Itpcc_com1 Itpcc_com1;
#endif /* __Itpcc_com1_FWD_DEFINED__ */

#ifndef __tpcc_com1_FWD_DEFINED__
#define __tpcc_com1_FWD_DEFINED__

#ifdef __cplusplus
typedef class tpcc_com1 tpcc_com1;
#else
typedef struct tpcc_com1 tpcc_com1;
#endif /* __cplusplus */

#endif /* __tpcc_com1_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 __Itpcc_com1_INTERFACE_DEFINED__
#define __Itpcc_com1_INTERFACE_DEFINED__

/* interface Itpcc_com1 */
/* [unique][helpstring][uuid][object] */

EXTERN_C const IID IID_Itpcc_com1;

#if defined(__cplusplus) && !defined(CINTERFACE)

    MIDL_INTERFACE("151287E4-C0DA-433C-A222-522B848087A5")
    Itpcc_com1 : 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_com1Vtbl
{
    BEGIN_INTERFACE

    HRESULT ( STDMETHODCALLTYPE *QueryInterface )(
        Itpcc_com1 * This,
        /* [in] */ REFIID riid,
        /* [iid_is][out] */ void **ppvObject);

    ULONG ( STDMETHODCALLTYPE *AddRef )(
        Itpcc_com1 * This);

    ULONG ( STDMETHODCALLTYPE *Release )(
        Itpcc_com1 * This);

    /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doStockLevel )(
        Itpcc_com1 * This,
        /* [in] */ INT *size,
        /* [size_is][size_is][out][in] */ UCHAR **buffer);

    /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doNewOrder )(
        Itpcc_com1 * This,
        /* [in] */ INT *size,
        /* [size_is][size_is][out][in] */ UCHAR **buffer);

    /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doPayment )(
        Itpcc_com1 * This,
        /* [in] */ INT *size,
        /* [size_is][size_is][out][in] */ UCHAR **buffer);

    /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doOrderStatus )(
        Itpcc_com1 * This,
        /* [in] */ INT *size,
        /* [size_is][size_is][out][in] */ UCHAR **buffer);

    /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doDBInfo )(
        Itpcc_com1 * This);

    /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
*doSetComplete )(
        Itpcc_com1 * This);

    END_INTERFACE
} Itpcc_com1Vtbl;

```

```

interface Itpcc_com1
{
    CONST_VTBL struct Itpcc_com1Vtbl *lpVtbl;
};

#ifdef COBJMACROS

#define Itpcc_com1_QueryInterface(This,riid,ppvObject) \
    (This)->lpVtbl->QueryInterface(This,riid,ppvObject)

#define Itpcc_com1_AddRef(This) \
    (This)->lpVtbl->AddRef(This)

#define Itpcc_com1_Release(This) \
    (This)->lpVtbl->Release(This)

#define Itpcc_com1_doStockLevel(This,size,buffer) \
    (This)->lpVtbl->doStockLevel(This,size,buffer)

#define Itpcc_com1_doNewOrder(This,size,buffer) \
    (This)->lpVtbl->doNewOrder(This,size,buffer)

#define Itpcc_com1_doPayment(This,size,buffer) \
    (This)->lpVtbl->doPayment(This,size,buffer)

#define Itpcc_com1_doOrderStatus(This,size,buffer) \
    (This)->lpVtbl->doOrderStatus(This,size,buffer)

#define Itpcc_com1_doDBInfo(This) \
    (This)->lpVtbl->doDBInfo(This)

#define Itpcc_com1_doSetComplete(This) \
    (This)->lpVtbl->doSetComplete(This)

#endif /* COBJMACROS */

#endif /* C style interface */

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com1_doStockLevel_Proxy(
    Itpcc_com1 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com1_doStockLevel_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com1_doNewOrder_Proxy(
    Itpcc_com1 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com1_doNewOrder_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *_pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

```

```

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com1_doPayment_Proxy(
    Itpcc_com1 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com1_doPayment_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer * _pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com1_doOrderStatus_Proxy(
    Itpcc_com1 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com1_doOrderStatus_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer * _pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com1_doDBInfo_Proxy(
    Itpcc_com1 * This);

void __RPC_STUB Itpcc_com1_doDBInfo_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer * _pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com1_doSetComplete_Proxy(
    Itpcc_com1 * This);

void __RPC_STUB Itpcc_com1_doSetComplete_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer * _pRpcChannelBuffer,
    PRPC_MESSAGE _pRpcMessage,
    DWORD *_pdwStubPhase);

#endif /* __Itpcc_com1_INTERFACE_DEFINED__ */

#ifndef __tpccCom1Lib_LIBRARY_DEFINED__
#define __tpccCom1Lib_LIBRARY_DEFINED__

/* library tpccCom1Lib */
/* [helpstring][version][uuid] */

EXTERN_C const IID LIBID_tpccCom1Lib;

EXTERN_C const CLSID CLSID_tpcc_com1;

#ifdef __cplusplus

```

```

class DECLSPEC_UUID("4549345B-0500-494F-AB02-
F7B4CB7AC1E2")
tpcc_com1;
#endif
#endif /* __tpccCom1Lib_LIBRARY_DEFINED__ */

```

/* Additional Prototypes for ALL interfaces */

/* end of Additional Prototypes */

```

#ifdef __cplusplus
}
#endif

#endif

```

tpccCom1/tpccCom1.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 Fri May 07 15:35:12 2004
*/

```

```

/* Compiler settings for .ItpccCom1.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( )

```

```

#ifdef !defined(_M_IA64) && !defined(_M_AMD64)

```

#pragma warning(disable: 4049) /* more than 64k source lines */

```

#ifdef __cplusplus
extern "C"{
#endif

```

```

#include <rpc.h>
#include <rpcndr.h>

```

```

#ifdef _MIDL_USE_GUIDDEF_

```

```

#ifndef INITGUID
#define INITGUID
#include <guiddef.h>
#undef INITGUID
#else
#include <guiddef.h>
#endif

```

```

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8) \
    DEFINE_GUID(name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8)

```

```

#else // !_MIDL_USE_GUIDDEF_

```

```

#ifndef __IID_DEFINED__

```

```

#define __IID_DEFINED__

typedef struct _IID
{
    unsigned long x;
    unsigned short s1;
    unsigned short s2;
    unsigned char c[8];
} IID;

#endif // __IID_DEFINED__

#ifndef CLSID_DEFINED
#define CLSID_DEFINED
typedef IID CLSID;
#endif // CLSID_DEFINED

#define
MIDL_DEFINE_GUID(type,name,l,w1,w2,b1,b2,b3,b4,b5,b6,b7,b8) \
    const type name = {l,w1,w2,{b1,b2,b3,b4,b5,b6,b7,b8}}

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
IID_Itpcc_com1,0x151287E4,0xC0DA,0x433C,0xA2,0x22,0x52,0x2
B,0x84,0x80,0x87,0xA5);

MIDL_DEFINE_GUID(IID,
LIBID_tpccCom1Lib,0x20D6DD23,0x491C,0x4F63,0xB2,0xE0,0xA6
,0x44,0xA4,0x83,0x53,0xB5);

MIDL_DEFINE_GUID(CLSID,
CLSID_tpcc_com1,0x4549345B,0x0500,0x494F,0xAB,0x02,0xF7,0x
B4,0xCB,0x7A,0xC1,0xE2);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif

#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/

```

tpccCom1/tpccCom1.idl

```

// tpccCom1.idl : IDL source for tpccCom1
//

// This file will be processed by the MIDL tool to
// produce the type library (tpccCom1.tlb) and marshalling code.

import "oidl.idl";
import "ocidl.idl";

[
    object,
    uuid(151287E4-C0DA-433C-A222-522B848087A5),
    // oleautomation,
    // nonextensible,
    helpstring("Itpcc_com1 Interface"),
    pointer_default(unique)
]
interface Itpcc_com1 : 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(20D6DD23-491C-4F63-B2E0-A644A48353B5),
    version(1.0),
    helpstring("tpccCom1 1.0 Type Library")
]
library tpccCom1Lib
{
    importlib("stdole2.tlb");
    [
        uuid(4549345B-0500-494F-AB02-F7B4CB7AC1E2),
        helpstring("tpcc_com1 Class")
    ]
    coclass tpcc_com1
    {
        [default] interface Itpcc_com1;
    };
};

```

tpccCom1/tpccCom1.p.c

/* this ALWAYS GENERATED file contains the proxy stub code */

```

/* File created by MIDL compiler version 6.00.0361 */
/* at Fri May 07 15:35:12 2004
*/
/* Compiler settings for .tpccCom1.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 "tpccCom1.h"

#define TYPE_FORMAT_STRING_SIZE 27
#define PROC_FORMAT_STRING_SIZE 229
#define TRANSMIT_AS_TABLE_SIZE 0
#define WIRE_MARSHAL_TABLE_SIZE 0

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,0x10,0
x48,0x60}},{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 ItpcCom1_ServerInfo;
extern const MIDL_STUBLESS_PROXY_INFO
ItpcCom1_ProxyInfo;

#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 doStockLevel */

        0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 2 */ NdrFcLong( 0x0 ),/* 0 */
/* 6 */ NdrFcShort( 0x3 ), /* 3 */
/* 8 */ NdrFcShort( 0x10 ),/* x86 Stack size/offset = 16 */
/* 10 */ NdrFcShort( 0x1c ),/* 28 */

```

```

/* 12 */ NdrFcShort( 0x8 ), /* 8 */
/* 14 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */
/* 16 */ 0x8, /* 8 */
        0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 18 */ NdrFcShort( 0x1 ), /* 1 */
/* 20 */ NdrFcShort( 0x1 ), /* 1 */
/* 22 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 24 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 26 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 28 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Parameter buffer */

/* 30 */ NdrFcShort( 0x201b ),/* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 32 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 34 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 36 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 38 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 40 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Procedure doNewOrder */

/* 42 */ 0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 44 */ NdrFcLong( 0x0 ),/* 0 */
/* 48 */ NdrFcShort( 0x4 ), /* 4 */
/* 50 */ NdrFcShort( 0x10 ),/* x86 Stack size/offset = 16 */
/* 52 */ NdrFcShort( 0x1c ),/* 28 */
/* 54 */ NdrFcShort( 0x8 ), /* 8 */
/* 56 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */
/* 58 */ 0x8, /* 8 */
        0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 60 */ NdrFcShort( 0x1 ), /* 1 */
/* 62 */ NdrFcShort( 0x1 ), /* 1 */
/* 64 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 66 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 68 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 70 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Parameter buffer */

/* 72 */ NdrFcShort( 0x201b ),/* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 74 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 76 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 78 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 80 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 82 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

```

```

/* Procedure doPayment */

/* 84 */ 0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 86 */ NdrFcLong( 0x0 ), /* 0 */
/* 90 */ NdrFcShort( 0x5 ), /* 5 */
/* 92 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 94 */ NdrFcShort( 0x1c ), /* 28 */
/* 96 */ NdrFcShort( 0x8 ), /* 8 */
/* 98 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */
/* 100 */ 0x8, /* 8 */
        0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 102 */ NdrFcShort( 0x1 ), /* 1 */
/* 104 */ NdrFcShort( 0x1 ), /* 1 */
/* 106 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 108 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 110 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 112 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Parameter buffer */

/* 114 */ NdrFcShort( 0x201b ), /* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 116 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 118 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 120 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 122 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 124 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Procedure doOrderStatus */

/* 126 */ 0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 128 */ NdrFcLong( 0x0 ), /* 0 */
/* 132 */ NdrFcShort( 0x6 ), /* 6 */
/* 134 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 136 */ NdrFcShort( 0x1c ), /* 28 */
/* 138 */ NdrFcShort( 0x8 ), /* 8 */
/* 140 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */
/* 142 */ 0x8, /* 8 */
        0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 144 */ NdrFcShort( 0x1 ), /* 1 */
/* 146 */ NdrFcShort( 0x1 ), /* 1 */
/* 148 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 150 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 152 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 154 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Parameter buffer */

/* 156 */ NdrFcShort( 0x201b ), /* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 158 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 160 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

```

```

/* Return value */

/* 162 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 164 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 166 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Procedure doDBInfo */

/* 168 */ 0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 170 */ NdrFcLong( 0x0 ), /* 0 */
/* 174 */ NdrFcShort( 0x7 ), /* 7 */
/* 176 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 178 */ NdrFcShort( 0x0 ), /* 0 */
/* 180 */ NdrFcShort( 0x8 ), /* 8 */
/* 182 */ 0x44, /* Oi2 Flags: has return, has ext, */
        0x1, /* 1 */
/* 184 */ 0x8, /* 8 */
        0x1, /* Ext Flags: new corr desc, */
/* 186 */ NdrFcShort( 0x0 ), /* 0 */
/* 188 */ NdrFcShort( 0x0 ), /* 0 */
/* 190 */ NdrFcShort( 0x0 ), /* 0 */

/* Return value */

/* 192 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 194 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 196 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

/* Procedure doSetComplete */

/* 198 */ 0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
/* 200 */ NdrFcLong( 0x0 ), /* 0 */
/* 204 */ NdrFcShort( 0x8 ), /* 8 */
/* 206 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 208 */ NdrFcShort( 0x0 ), /* 0 */
/* 210 */ NdrFcShort( 0x8 ), /* 8 */
/* 212 */ 0x44, /* Oi2 Flags: has return, has ext, */
        0x1, /* 1 */
/* 214 */ 0x8, /* 8 */
        0x1, /* Ext Flags: new corr desc, */
/* 216 */ NdrFcShort( 0x0 ), /* 0 */
/* 218 */ NdrFcShort( 0x0 ), /* 0 */
/* 220 */ NdrFcShort( 0x0 ), /* 0 */

/* Return value */

/* 222 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 224 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 226 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

        0x0
    }
};

static const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString =
{
    0,
    {
        NdrFcShort( 0x0 ), /* 0 */
/* 2 */
        0x11, 0x8, /* FC_RP [simple_pointer] */
/* 4 */ 0x8, /* FC_LONG */
        0x5c, /* FC_PAD */
/* 6 */
        0x11, 0x14, /* FC_RP [allocated_on_stack] [pointer_deref] */

```



```

/* 8 */ NdrFcShort( 0x2 ), /* Offset= 2 (10) */
/* 10 */
    0x13, 0x0, /* FC_OP */
/* 12 */ NdrFcShort( 0x2 ), /* Offset= 2 (14) */
/* 14 */
    0x1b, /* FC_CARRAY */
    0x0, /* 0 */
/* 16 */ NdrFcShort( 0x1 ), /* 1 */
/* 18 */ 0x28, /* Corr desc: parameter, FC_LONG */
    0x54, /* FC_DEREFERENCE */
/* 20 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 22 */ NdrFcShort( 0x1 ), /* Corr flags: early, */
/* 24 */ 0x2, /* FC_CHAR */
    0x5b, /* FC_END */

    0x0
}
};

/* Object interface: IUnknown, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x
00,0x00,0x46}} */

/* Object interface: Itpcc_com1, ver. 0.0,

GUID={0x151287E4,0xC0DA,0x433C,{0xA2,0x22,0x52,0x2B,0x84,
0x80,0x87,0xA5}} */

#pragma code_seg(".orpc")
static const unsigned short Itpcc_com1_FormatStringOffsetTable[] =
{
    0,
    42,
    84,
    126,
    168,
    198
};

static const MIDL_STUBLESS_PROXY_INFO
Itpcc_com1_ProxyInfo =
{
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &Itpcc_com1_FormatStringOffsetTable[-3],
    0,
    0,
    0
};

static const MIDL_SERVER_INFO Itpcc_com1_ServerInfo =
{
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &Itpcc_com1_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0
};

CINTERFACE_PROXY_VTABLE(9) _Itpcc_com1ProxyVtbl =
{
    &Itpcc_com1_ProxyInfo,
    &IID_Itpcc_com1,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    (void *) (INT_PTR) -1 /* Itpcc_com1::doStockLevel */ ,

```

```

(void *) (INT_PTR) -1 /* Itpcc_com1::doNewOrder */ ,
(void *) (INT_PTR) -1 /* Itpcc_com1::doPayment */ ,
(void *) (INT_PTR) -1 /* Itpcc_com1::doOrderStatus */ ,
(void *) (INT_PTR) -1 /* Itpcc_com1::doDBInfo */ ,
(void *) (INT_PTR) -1 /* Itpcc_com1::doSetComplete */
};

const CInterfaceStubVtbl _Itpcc_com1StubVtbl =
{
    &IID_Itpcc_com1,
    &Itpcc_com1_ServerInfo,
    9,
    0, /* pure interpreted */
    CStdStubBuffer_METHODS
};

static const MIDL_STUB_DESC Object_StubDesc =
{
    0,
    NdrOleAllocate,
    NdrOleFree,
    0,
    0,
    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,
    0,
    0, /* notify & notify_flag routine table */
    0x1, /* MIDL flag */
    0, /* cs routines */
    0, /* proxy/server info */
    0 /* Reserved5 */
};

const CInterfaceProxyVtbl * _tpccCom1_ProxyVtblList[] =
{
    {
        ( CInterfaceProxyVtbl *) &_Itpcc_com1ProxyVtbl,
        0
    }
};

const CInterfaceStubVtbl * _tpccCom1_StubVtblList[] =
{
    {
        ( CInterfaceStubVtbl *) &_Itpcc_com1StubVtbl,
        0
    }
};

PCInterfaceName const _tpccCom1_InterfaceNamesList[] =
{
    "Itpcc_com1",
    0
};

#define _tpccCom1_CHECK_IID(n)
IID_GENERIC_CHECK_IID( _tpccCom1, pIID, n)

int __stdcall _tpccCom1_IID_Lookup( const IID * pIID, int * pIndex )
{
    if(!_tpccCom1_CHECK_IID(0))
    {
        *pIndex = 0;
        return 1;
    }

    return 0;
}

```

```

}

const ExtendedProxyFileInfo tpccCom1_ProxyFileInfo =
{
    (PCInterfaceProxyVtblList *) & _tpccCom1_ProxyVtblList,
    (PCInterfaceStubVtblList *) & _tpccCom1_StubVtblList,
    (const PCInterfaceName *) & _tpccCom1_InterfaceNamesList,
    0, // no delegation
    & _tpccCom1_IID_Lookup,
    1,
    2,
    0, /* table of [async_uuid] interfaces */
    0, /* Filler1 */
    0, /* Filler2 */
    0 /* Filler3 */
};
#endif _MSC_VER >= 1200
#pragma warning(pop)
#endif

```

```

#endif /* !defined(_M_IA64) && !defined(_M_AMD64) */

```

tpccCom1/tpccCom1.rc

```

// Microsoft Visual C++ generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "winres.h"

//
// English (U.S.) resources
//

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifndef _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 \"\"tpccCom1.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", "tpccCom1.dll"
            VALUE "OriginalFilename", "tpccCom1.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_TPCCCOM1      REGISTRY      "tpccCom1.rgs"
IDR_TPCC_COM1     REGISTRY      "tpcc_com1.rgs"

```

```

//
// String Table
//

STRINGTABLE
BEGIN
    IDS_PROJNAME      "tpccCom1"
END

```

```

#endif // English (U.S.) resources
//

```

```

#ifdef APSTUDIO_INVOKED
//
// Generated from the TEXTINCLUDE 3 resource.
//
1 TYPELIB "tpccCom1.tlb"

```

```

////////////////////////////////////
#endif // not APSTUDIO_INVOKED

```

tpccCom1/tpcc_com1.rgs

```

HKCR
{
    tpccCom1.tpcc_com1.1 = s 'tpcc_com1 Class'
    {
        CLSID = s '{4549345B-0500-494F-AB02-F7B4CB7AC1E2}'
    }
    tpccCom1.tpcc_com1 = s 'tpcc_com1 Class'
    {
        CLSID = s '{4549345B-0500-494F-AB02-F7B4CB7AC1E2}'
        CurVer = s 'tpccCom1.tpcc_com1.1'
    }
    NoRemove CLSID
    {
        ForceRemove {4549345B-0500-494F-AB02-F7B4CB7AC1E2} =
s 'tpcc_com1 Class'
        {
            ProgID = s 'tpccCom1.tpcc_com1.1'
            VersionIndependentProgID = s 'tpccCom1.tpcc_com1'
            InprocServer32 = s '%MODULE%'
            {
                val ThreadingModel = s 'Both'
            }
            val AppID = s '%APPID%'
            'TypeLib' = s '{20D6DD23-491C-4F63-B2E0-A644A48353B5}'
        }
    }
}

```

tpccCom1/tpccCom1.rgs

```

HKCR
{
    NoRemove AppID
    {
        '%APPID%' = s 'tpccCom1'
        'tpccCom1.DLL'
        {
            val AppID = s '%APPID%'
        }
    }
}

```

tpccCom50/dlldata.c

```

/*****

```

DllData file -- generated by MIDL compiler

DO NOT ALTER THIS FILE

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

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

```

*****/

```

```

#include <rpcproxy.h>

```

```

#ifdef __cplusplus
extern "C" {
#endif

```

```

EXTERN_PROXY_FILE( tpccCom50 )

```

```

PROXYFILE_LIST_START
/* Start of list */
REFERENCE_PROXY_FILE( tpccCom50 ),
/* End of list */
PROXYFILE_LIST_END

```

```

DLLDATA_ROUTINES( aProxyFileList, GET_DLL_CLSID )

```

```

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

```

```

/* end of generated dlldata file */

```

tpccCom50/dlldata.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 "tpccCom50_p.c"

```

```

#endif // _MERGE_PROXYSTUB

```

tpccCom50/dlldata.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

```

tpccCom50/resource.h

```

//{{NO_DEPENDENCIES}}
// Microsoft Visual C++ generated include file.
// Used by tpccCom50.rc
//
#define IDS_PROJNAME            100
#define IDR_TPCCCOM50          101
#define IDR_TPCC_COM50         102

```

```

// Next default values for new objects
//

```

```

#ifdef APSTUDIO_INVOKED
#ifdef 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

tpccCom50/stdafx.cpp

// stdafx.cpp : source file that includes just the standard includes
// tpccCom50.pch will be the pre-compiled header
// stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

tpccCom50/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;

tpccCom50/tpcc_com50.cpp

// tpcc_com50.cpp : Implementation of Ctpcc_com50

#include "stdafx.h"
#include "tpcc_com50.h"

#include "..\tpcc\api\tpcc.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_com50
HRESULT Ctpcc_com50::Activate()
{
    HRESULT hr = GetObjectContext(&m_spObjectContext);
    if (SUCCEEDED(hr))
    {
        DEBUGMSG("Object assigned to thread."<<endl);
        return S_OK;
    }
    return hr;
}

BOOL Ctpcc_com50::CanBePooled()
{
    DEBUGMSG("CanBePooled() returning true"<<endl);
    return TRUE;
}

void Ctpcc_com50::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_com50::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_com50::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);

    do_stok(stok,connectHandle);

    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_com50::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_com50::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_com50::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);

    do_ords(ords,connectHandle);

    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_com50::doDBInfo(void)
{
    //DUMMY FUNCTION TO ALLOW ISAPI TO CALL COM
    FUNCTION DURING LOGIN SO CONNECTIONS
    //TO DATABASE WILL BE SETUP.....I HOPE.
    DEBUGMSG("Stub function to warm object pool"<<endl);

    return S_OK;
}

/*
*****
** Name      : loadLibrary
** Description :
**          Function loads apprioate db library based on
**          registry setting
** Parameters:
** Returns    :
**          int - return code
** Comments   :
**
*****
*/

Ctpcc_com50::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
{
    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,"disconnect_db")
) == NULL )
    return ERR_DISCONNECT_ADDRESS_NOT_FOUND;
DEBUGMSG("do_disconnect
address:"<<DEBUGADDRESS(do_disconnect)<<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
address:"<<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 registry
**           dbName, dbUserName, and dbUserPassword
*****
*/

Ctpcc_com50::readRegistry()
{
    //open registry key
    HKEY registryKey;
    DWORD regType;
    char value[MAX_STRING_LEN];
    DWORD regValue;
    DWORD regValueSize = MAX_STRING_LEN;

    DEBUGMSG("Entered readRegistry(), opening key:"<<
REGISTRY_SUB_KEY <<endl);
    //open up registry key
    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTRY_SUB_
KEY,0,KEY_READ,&registryKey) == ERROR_SUCCESS)
    {
        DEBUGMSG(REGISTRY_SUB_KEY<<" open, getting
database type from key"<<endl);
        regValueSize = sizeof(value);
        if
(RegQueryValueEx(registryKey,DB_TYPE,0,&regType,(BYTE *)
&value,&regValueSize)== ERROR_SUCCESS )
            strcpy(dbType,value);
        DEBUGMSG("Database type:"<<dbType<<" from registry
key."<<endl);

        DEBUGMSG("Getting database name from registry
key."<<endl);
        regValueSize = sizeof(value);
        if
(RegQueryValueEx(registryKey,DB_NAME,0,&regType,(BYTE *)
&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(registryKey,NULL_DB,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
            nullDB = regValue;
        DEBUGMSG("Null database flag:"<<nullDB<<endl);

        return OK;
    }

    DEBUGMSG("Error, unable to open registry key."<<endl);
    return ERR_UNABLE_TO_OPEN_REG;
}

/*
*****
** Name      : connectDB
** Description :
**      Function connects to the db
** Parameters:
** Returns    :
**      int - return code
** Comments   :
**
*****
*/
Ctpcc_com50::connectDB()
{
    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)
        {
            DEBUGMSG("Setting Context handle in use to true"<<endl);
            connectHandleInUse = 1;
            connected = do_connection(dbName,&connectHandle);
            if(connected != OK)
            {
                DEBUGMSG("Object do_connect failed,
rc:"<<connected<<endl);
                ERRORMSG("Object do_connect failed,
rc:"<<connected<<endl);
                return connected;
            }
            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;
}

```

tpccCom50/tpccCom50.cpp

```

// tpccCom50.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 "tpccCom50.h"
#include "dlldata.h"

class CtpccCom50Module : public CAtlDllModuleT<
CtpccCom50Module >
{
public :
    DECLARE_LIBID(LIBID_tpccCom50Lib)
    DECLARE_REGISTRY_APPID_RESOURCEID(IDR_TPCCCOM5
0, "{C5FF7862-C99E-4A03-828C-2513A1B8A90D}")
};

CtpccCom50Module _AtlModule;

// 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;
    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;
}
```

tpccCom50/tpccCom50.def

; tpccCom50.def : Declares the module parameters.

```
LIBRARY "tpccCom50.DLL"
```

```
EXPORTS
    DllCanUnloadNow PRIVATE
    DllGetClassObject PRIVATE
    DllRegisterServer PRIVATE
    DllUnregisterServer PRIVATE
```

tpccCom50/tpcc_com50.h

// tpcc_com50.h : Declaration of the Ctpcc_com50

```
#pragma once
#include "tpccCom50.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 int comServerID = 0;
static ofstream debugStream;
static ofstream errorStream;
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);
```

```
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_com50
class ATL_NO_VTABLE Ctpcc_com50 :
public CComObjectRootEx<CComMultiThreadModel>,
public IObjectControl,
public CComCoClass<Ctpcc_com50, &CLSID_tpcc_com50>,
public Itpcc_com50
{
public:
    Ctpcc_com50()
    {
        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_debug.txt");
                debugStream.rdbuf( )->open(comLogFile,ios_base::in |
ios_base::out | ios_base::app);

                debugFileOpen = 1;
            }

            //open error log file
            if(!errorFileOpen)
            {
                InitializeCriticalSection(&errorMutex);

                char errorLogFile[128];

                sprintf(errorLogFile,"C:\\inetpub\\wwwroot\\tpcc\\comLog_err.txt");

                errorStream.rdbuf( )->open(errorLogFile,ios_base::in |
ios_base::out | ios_base::app);

                errorFileOpen=1;
            }

            //get registry values
            if((rc = readRegistry()) != OK)
            {
                ERRORMSG("Unable to open registry key " <<
REGISTRY_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_COM50)

BEGIN_COM_MAP(Ctpcc_com50)
    COM_INTERFACE_ENTRY(Itpcc_com50)
    COM_INTERFACE_ENTRY(IObjControl)
END_COM_MAP()

// IObjControl
public:
    STDMETHODCALLTYPE(Activate)();
    STDMETHODCALLTYPE_(BOOL, CanBePooled)();
    STDMETHODCALLTYPE_(void, Deactivate)();
    CComPtr<IObjContext> m_spObjectContext;

// Itpcc_com50
public:
    STDMETHODCALLTYPE(doStockLevel)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doNewOrder)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doPayment)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doOrderStatus)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doDBInfo)(void);
    STDMETHODCALLTYPE(doSetComplete)(void);

    int connected;
    int connectHandleInUse;

private:
    //db2 specific context
    void *connectHandle;
    int loadLibrary();
    int readRegistry();
    int connectDB();

};

OBJECT_ENTRY_AUTO(__uuidof(tpcc_com50), Ctpcc_com50)

```

tpccCom50/tpccCom50.h

```

/* this ALWAYS GENERATED file contains the definitions for the
interfaces */

/* File created by MIDL compiler version 6.00.0361 */
/* at Sat May 08 22:55:51 2004
*/
/* Compiler settings for .ItpccCom50.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 __tpccCom50_h__
#define __tpccCom50_h__

#if defined(_MSC_VER) && (_MSC_VER >= 1020)
#pragma once
#endif

/* Forward Declarations */

#ifndef __Itpcc_com50_FWD_DEFINED__
#define __Itpcc_com50_FWD_DEFINED__
typedef interface Itpcc_com50 Itpcc_com50;
#endif /* __Itpcc_com50_FWD_DEFINED__ */

#ifndef __tpcc_com50_FWD_DEFINED__
#define __tpcc_com50_FWD_DEFINED__

#ifdef __cplusplus
typedef class tpcc_com50 tpcc_com50;
#else
typedef struct tpcc_com50 tpcc_com50;
#endif /* __cplusplus */

#endif /* __tpcc_com50_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 * );

#ifdef __Itpcc_com50_INTERFACE_DEFINED__
#define __Itpcc_com50_INTERFACE_DEFINED__

/* interface Itpcc_com50 */
/* [unique][helpstring][uuid][object] */

EXTERN_C const IID IID_Itpcc_com50;

#ifdef __cplusplus && !defined(CINTERFACE)

    MIDL_INTERFACE("DBA619F6-1BDE-4E34-800E-3723C4BA95C1")
    Itpcc_com50 : 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_com50Vtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE *QueryInterface)(
            Itpcc_com50 * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void **ppvObject);

        ULONG ( STDMETHODCALLTYPE *AddRef )(
            Itpcc_com50 * This);

        ULONG ( STDMETHODCALLTYPE *Release )(
            Itpcc_com50 * This);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
doStockLevel )(
            Itpcc_com50 * This,
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer);

```

```

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
doNewOrder )(
            Itpcc_com50 * This,
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
doPayment )(
            Itpcc_com50 * This,
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
doOrderStatus )(
            Itpcc_com50 * This,
            /* [in] */ INT *size,
            /* [size_is][size_is][out][in] */ UCHAR **buffer);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
doDBInfo )(
            Itpcc_com50 * This);

        /* [helpstring] */ HRESULT ( STDMETHODCALLTYPE
doSetComplete )(
            Itpcc_com50 * This);

        END_INTERFACE
    } Itpcc_com50Vtbl;

    interface Itpcc_com50
    {
        CONST_VTBL struct Itpcc_com50Vtbl *lpVtbl;
    };

#ifdef COBJMACROS

#define Itpcc_com50_QueryInterface(This,riid,ppvObject) \
    (This->lpVtbl->QueryInterface(This,riid,ppvObject)

#define Itpcc_com50_AddRef(This) \
    (This->lpVtbl->AddRef(This)

#define Itpcc_com50_Release(This) \
    (This->lpVtbl->Release(This)

#define Itpcc_com50_doStockLevel(This,size,buffer) \
    (This->lpVtbl->doStockLevel(This,size,buffer)

#define Itpcc_com50_doNewOrder(This,size,buffer) \
    (This->lpVtbl->doNewOrder(This,size,buffer)

#define Itpcc_com50_doPayment(This,size,buffer) \
    (This->lpVtbl->doPayment(This,size,buffer)

#define Itpcc_com50_doOrderStatus(This,size,buffer) \
    (This->lpVtbl->doOrderStatus(This,size,buffer)

#define Itpcc_com50_doDBInfo(This) \
    (This->lpVtbl->doDBInfo(This)

#define Itpcc_com50_doSetComplete(This) \
    (This->lpVtbl->doSetComplete(This)

#endif /* COBJMACROS */

#endif /* C style interface */

```

```

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com50_doStockLevel_Proxy(
    Itpcc_com50 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com50_doStockLevel_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *pRpcChannelBuffer,
    PRPC_MESSAGE_pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com50_doNewOrder_Proxy(
    Itpcc_com50 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com50_doNewOrder_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *pRpcChannelBuffer,
    PRPC_MESSAGE_pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com50_doPayment_Proxy(
    Itpcc_com50 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com50_doPayment_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *pRpcChannelBuffer,
    PRPC_MESSAGE_pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com50_doOrderStatus_Proxy(
    Itpcc_com50 * This,
    /* [in] */ INT *size,
    /* [size_is][size_is][out][in] */ UCHAR **buffer);

void __RPC_STUB Itpcc_com50_doOrderStatus_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *pRpcChannelBuffer,
    PRPC_MESSAGE_pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com50_doDBInfo_Proxy(
    Itpcc_com50 * This);

void __RPC_STUB Itpcc_com50_doDBInfo_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *pRpcChannelBuffer,
    PRPC_MESSAGE_pRpcMessage,
    DWORD *_pdwStubPhase);

/* [helpstring] */ HRESULT STDMETHODCALLTYPE
Itpcc_com50_doSetComplete_Proxy(

```

```

    Itpcc_com50 * This);

void __RPC_STUB Itpcc_com50_doSetComplete_Stub(
    IRpcStubBuffer *This,
    IRpcChannelBuffer *pRpcChannelBuffer,
    PRPC_MESSAGE_pRpcMessage,
    DWORD *_pdwStubPhase);

```

```

#endif /* __Itpcc_com50_INTERFACE_DEFINED__ */

```

```

#ifndef __tpccCom50Lib_LIBRARY_DEFINED__
#define __tpccCom50Lib_LIBRARY_DEFINED__

```

```

/* library tpccCom50Lib */
/* [helpstring][version][uuid] */

```

```

EXTERN_C const IID LIBID_tpccCom50Lib;

```

```

EXTERN_C const CLSID CLSID_tpcc_com50;

```

```

#ifdef __cplusplus

```

```

class DECLSPEC_UUID("76F680E9-0C71-4AF3-B9F1-
6B0A3AC4E1B8")
tpcc_com50;
#endif
#endif /* __tpccCom50Lib_LIBRARY_DEFINED__ */

```

```

/* Additional Prototypes for ALL interfaces */

```

```

/* end of Additional Prototypes */

```

```

#ifdef __cplusplus
}
#endif

#endif

```

tpccCom50/tpccCom50_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 May 08 22:55:51 2004
*/

```

```

/* Compiler settings for .\tpccCom50.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_Itpcc_com50,0xDBA619F6,0x1BDE,0x4E34,0x80,0x0E,0x37,0x
23,0xC4,0xBA,0x95,0xC1);

MIDL_DEFINE_GUID(IID,
LIBID_tpccCom50Lib,0x7BD99942,0x8C52,0x4AEE,0x9B,0x82,0xB
E,0xCC,0x2C,0x7B,0x12,0x07);

MIDL_DEFINE_GUID(CLSID,
CLSID_tpcc_com50,0x76F680E9,0x0C71,0x4AF3,0xB9,0xF1,0x6B,
0x0A,0x3A,0xC4,0xE1,0xB8);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif

```

```

#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/

```

tpccCom50/tpccCom50.idl

```

// tpccCom50.idl : IDL source for tpccCom50
//

// This file will be processed by the MIDL tool to
// produce the type library (tpccCom50.tlb) and marshalling code.

import "oidl.idl";
import "ocidl.idl";

[
    object,
    uuid(DBA619F6-1BDE-4E34-800E-3723C4BA95C1),
    // oleautomation,
    // nonextensible,
    helpstring("Itpcc_com50 Interface"),
    pointer_default(unique)
]
interface Itpcc_com50 : 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(7BD99942-8C52-4AEE-9B82-BECC2C7B1207),
    version(1.0),
    helpstring("tpccCom50 1.0 Type Library")
]
library tpccCom50Lib
{
    importlib("stdole2.tlb");
    [
        uuid(76F680E9-0C71-4AF3-B9F1-6B0A3AC4E1B8),
        helpstring("tpcc_com50 Class")
    ]
    coclass tpcc_com50
    {
        [default] interface Itpcc_com50;
    };
};

```

tpccCom50/tpccCom50.p.c

```

/* this ALWAYS GENERATED file contains the proxy stub code */

```

```

/* File created by MIDL compiler version 6.00.0361 */
/* at Sat May 08 22:55:51 2004
*/
/* Compiler settings for .tpccCom50.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 "tpccCom50.h"

#define TYPE_FORMAT_STRING_SIZE 27
#define PROC_FORMAT_STRING_SIZE 229
#define TRANSMIT_AS_TABLE_SIZE 0
#define WIRE_MARSHAL_TABLE_SIZE 0

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,0x10,0
x48,0x60}},{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 Itppcc_com50_ServerInfo;
extern const MIDL_STUBLESS_PROXY_INFO
Itppcc_com50_ProxyInfo;

#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 doStockLevel */

        0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
        /* 2 */ NdrFcLong( 0x0 ), /* 0 */
        /* 6 */ NdrFcShort( 0x3 ), /* 3 */
        /* 8 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
        /* 10 */ NdrFcShort( 0x1c ), /* 28 */
        /* 12 */ NdrFcShort( 0x8 ), /* 8 */
        /* 14 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */
        /* 16 */ 0x8, /* 8 */
        0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
        /* 18 */ NdrFcShort( 0x1 ), /* 1 */
        /* 20 */ NdrFcShort( 0x1 ), /* 1 */
        /* 22 */ NdrFcShort( 0x0 ), /* 0 */

        /* Parameter size */

        /* 24 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
        /* 26 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
        /* 28 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

        /* Parameter buffer */

        /* 30 */ NdrFcShort( 0x201b ), /* Flags: must size, must free, in, out,
srv alloc size=8 */
        /* 32 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
        /* 34 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

        /* Return value */

        /* 36 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
        /* 38 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
        /* 40 */ 0x8, /* FC_LONG */
        0x0, /* 0 */

        /* Procedure doNewOrder */

        /* 42 */ 0x33, /* FC_AUTO_HANDLE */
        0x6c, /* Old Flags: object, Oi2 */
        /* 44 */ NdrFcLong( 0x0 ), /* 0 */
        /* 48 */ NdrFcShort( 0x4 ), /* 4 */
        /* 50 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
        /* 52 */ NdrFcShort( 0x1c ), /* 28 */
        /* 54 */ NdrFcShort( 0x8 ), /* 8 */
        /* 56 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
        0x3, /* 3 */

```

```

/* 58 */ 0x8, /* 8 */
    0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 60 */ NdrFcShort( 0x1 ), /* 1 */
/* 62 */ NdrFcShort( 0x1 ), /* 1 */
/* 64 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 66 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 68 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 70 */ 0x8, /* FC_LONG */
    0x0, /* 0 */

/* Parameter buffer */

/* 72 */ NdrFcShort( 0x201b ), /* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 74 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 76 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 78 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 80 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 82 */ 0x8, /* FC_LONG */
    0x0, /* 0 */

/* Procedure doPayment */

/* 84 */ 0x33, /* FC_AUTO_HANDLE */
    0x6c, /* Old Flags: object, Oi2 */
/* 86 */ NdrFcLong( 0x0 ), /* 0 */
/* 90 */ NdrFcShort( 0x5 ), /* 5 */
/* 92 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 94 */ NdrFcShort( 0x1c ), /* 28 */
/* 96 */ NdrFcShort( 0x8 ), /* 8 */
/* 98 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
    0x3, /* 3 */
/* 100 */ 0x8, /* 8 */
    0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 102 */ NdrFcShort( 0x1 ), /* 1 */
/* 104 */ NdrFcShort( 0x1 ), /* 1 */
/* 106 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 108 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 110 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 112 */ 0x8, /* FC_LONG */
    0x0, /* 0 */

/* Parameter buffer */

/* 114 */ NdrFcShort( 0x201b ), /* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 116 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 118 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 120 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 122 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 124 */ 0x8, /* FC_LONG */
    0x0, /* 0 */

/* Procedure doOrderStatus */

/* 126 */ 0x33, /* FC_AUTO_HANDLE */
    0x6c, /* Old Flags: object, Oi2 */

```

```

/* 128 */ NdrFcLong( 0x0 ), /* 0 */
/* 132 */ NdrFcShort( 0x6 ), /* 6 */
/* 134 */ NdrFcShort( 0x10 ), /* x86 Stack size/offset = 16 */
/* 136 */ NdrFcShort( 0x1c ), /* 28 */
/* 138 */ NdrFcShort( 0x8 ), /* 8 */
/* 140 */ 0x47, /* Oi2 Flags: srv must size, clt must size, has
return, has ext, */
    0x3, /* 3 */
/* 142 */ 0x8, /* 8 */
    0x7, /* Ext Flags: new corr desc, clt corr check, srv corr
check, */
/* 144 */ NdrFcShort( 0x1 ), /* 1 */
/* 146 */ NdrFcShort( 0x1 ), /* 1 */
/* 148 */ NdrFcShort( 0x0 ), /* 0 */

/* Parameter size */

/* 150 */ NdrFcShort( 0x148 ), /* Flags: in, base type, simple ref, */
/* 152 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 154 */ 0x8, /* FC_LONG */
    0x0, /* 0 */

/* Parameter buffer */

/* 156 */ NdrFcShort( 0x201b ), /* Flags: must size, must free, in, out,
srv alloc size=8 */
/* 158 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 160 */ NdrFcShort( 0x6 ), /* Type Offset=6 */

/* Return value */

/* 162 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 164 */ NdrFcShort( 0xc ), /* x86 Stack size/offset = 12 */
/* 166 */ 0x8, /* FC_LONG */
    0x0, /* 0 */

/* Procedure doDBInfo */

/* 168 */ 0x33, /* FC_AUTO_HANDLE */
    0x6c, /* Old Flags: object, Oi2 */
/* 170 */ NdrFcLong( 0x0 ), /* 0 */
/* 174 */ NdrFcShort( 0x7 ), /* 7 */
/* 176 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 178 */ NdrFcShort( 0x0 ), /* 0 */
/* 180 */ NdrFcShort( 0x8 ), /* 8 */
/* 182 */ 0x44, /* Oi2 Flags: has return, has ext, */
    0x1, /* 1 */
/* 184 */ 0x8, /* 8 */
    0x1, /* Ext Flags: new corr desc, */
/* 186 */ NdrFcShort( 0x0 ), /* 0 */
/* 188 */ NdrFcShort( 0x0 ), /* 0 */
/* 190 */ NdrFcShort( 0x0 ), /* 0 */

/* Return value */

/* 192 */ NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 194 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 196 */ 0x8, /* FC_LONG */
    0x0, /* 0 */

/* Procedure doSetComplete */

/* 198 */ 0x33, /* FC_AUTO_HANDLE */
    0x6c, /* Old Flags: object, Oi2 */
/* 200 */ NdrFcLong( 0x0 ), /* 0 */
/* 204 */ NdrFcShort( 0x8 ), /* 8 */
/* 206 */ NdrFcShort( 0x8 ), /* x86 Stack size/offset = 8 */
/* 208 */ NdrFcShort( 0x0 ), /* 0 */
/* 210 */ NdrFcShort( 0x8 ), /* 8 */
/* 212 */ 0x44, /* Oi2 Flags: has return, has ext, */
    0x1, /* 1 */
/* 214 */ 0x8, /* 8 */

```

```

        0x1, /* Ext Flags: new corr desc, */
/* 216 */NdrFcShort( 0x0 ), /* 0 */
/* 218 */NdrFcShort( 0x0 ), /* 0 */
/* 220 */NdrFcShort( 0x0 ), /* 0 */

/* Return value */

/* 222 */NdrFcShort( 0x70 ), /* Flags: out, return, base type, */
/* 224 */NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 226 */0x8, /* FC_LONG */
0x0, /* 0 */

0x0
}
};

static const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString =
{
    0,
    {
        NdrFcShort( 0x0 ), /* 0 */
/* 2 */
0x11, 0x8, /* FC_RP [simple_pointer] */
/* 4 */ 0x8, /* FC_LONG */
0x5c, /* FC_PAD */
/* 6 */
0x11, 0x14, /* FC_RP [allocated_on_stack] [pointer_deref] */
/* 8 */ NdrFcShort( 0x2 ), /* Offset= 2 (10) */
/* 10 */
0x13, 0x0, /* FC_OP */
/* 12 */ NdrFcShort( 0x2 ), /* Offset= 2 (14) */
/* 14 */
0x1b, /* FC_CARRAY */
0x0, /* 0 */
/* 16 */ NdrFcShort( 0x1 ), /* 1 */
/* 18 */ 0x28, /* Corr desc: parameter, FC_LONG */
0x54, /* FC_DEREFERENCE */
/* 20 */ NdrFcShort( 0x4 ), /* x86 Stack size/offset = 4 */
/* 22 */ NdrFcShort( 0x1 ), /* Corr flags: early, */
/* 24 */ 0x2, /* FC_CHAR */
0x5b, /* FC_END */

0x0
}
};

/* Object interface: IUnknown, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0xC0,0x00,0x00,0x00,0x00,0x00,0x46}} */

/* Object interface: Itpcc_com50, ver. 0.0,

GUID={0xDBA619F6,0x1BDE,0x4E34,{0x80,0x0E,0x37,0x23,0xC4,0xBA,0x95,0xC1}} */

#pragma code_seg(".orpc")
static const unsigned short Itpcc_com50_FormatStringOffsetTable[]
=
{
    0,
    42,
    84,
    126,
    168,
    198
};

```

```

static const MIDL_STUBLESS_PROXY_INFO
Itpcc_com50_ProxyInfo =
{
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &Itpcc_com50_FormatStringOffsetTable[-3],
    0,
    0,
    0
};

static const MIDL_SERVER_INFO Itpcc_com50_ServerInfo =
{
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &Itpcc_com50_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0
};

CINTERFACE_PROXY_VTABLE(9) _Itpcc_com50ProxyVtbl =
{
    &Itpcc_com50_ProxyInfo,
    &IID_Itpcc_com50,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    (void *) (INT_PTR) -1 /* Itpcc_com50::doStockLevel */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com50::doNewOrder */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com50::doPayment */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com50::doOrderStatus */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com50::doDBInfo */ ,
    (void *) (INT_PTR) -1 /* Itpcc_com50::doSetComplete */
};

const CInterfaceStubVtbl _Itpcc_com50StubVtbl =
{
    &IID_Itpcc_com50,
    &Itpcc_com50_ServerInfo,
    9,
    0, /* pure interpreted */
    CStdStubBuffer_METHODS
};

static const MIDL_STUB_DESC Object_StubDesc =
{
    0,
    NdrOleAllocate,
    NdrOleFree,
    0,
    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,
    0,
    0, /* notify & notify_flag routine table */
    0x1, /* MIDL flag */
    0, /* cs routines */
    0, /* proxy/server info */
    0 /* Reserved5 */
};

const CInterfaceProxyVtbl * _tpccCom50_ProxyVtblList[] =
{

```



```

    ( CInterfaceProxyVtbl *) &_tpcc_com50ProxyVtbl,
    0
};

const CInterfaceStubVtbl * _tpccCom50_StubVtblList[] =
{
    ( CInterfaceStubVtbl *) &_tpcc_com50StubVtbl,
    0
};

PCInterfaceName const _tpccCom50_InterfaceNamesList[] =
{
    "Itpcc_com50",
    0
};

#define _tpccCom50_CHECK_IID(n)
IID_GENERIC_CHECK_IID( _tpccCom50, pIID, n)

int __stdcall _tpccCom50_IID_Lookup( const IID * pIID, int * pIndex )
{
    if(!_tpccCom50_CHECK_IID(0))
    {
        *pIndex = 0;
        return 1;
    }

    return 0;
}

const ExtendedProxyFileInfo tpccCom50_ProxyFileInfo =
{
    (PCInterfaceProxyVtblList *) &_tpccCom50_ProxyVtblList,
    (PCInterfaceStubVtblList *) &_tpccCom50_StubVtblList,
    (const PCInterfaceName *) &_tpccCom50_InterfaceNamesList,
    0, // no delegation
    &_tpccCom50_IID_Lookup,
    1,
    2,
    0, /* table of [async_uuid] interfaces */
    0, /* Filler1 */
    0, /* Filler2 */
    0 /* Filler3 */
};

#if _MSC_VER >= 1200
#pragma warning(pop)
#endif

#endif /* !defined(_M_IA64) && !defined(_M_AMD64)*/

```

tpccCom50/tpccCom50.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

```

```

#ifdef 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 \"\"tpccCom50.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", "tpccCom50.dll"
            VALUE "OriginalFilename", "tpccCom50.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_TPCCCOM50      REGISTRY      "tpccCom50.rgs"
IDR_TPCC_COM50     REGISTRY      "tpcc_com50.rgs"

////////////////////////////////////
//
// String Table
//

STRINGTABLE
BEGIN
    IDS_PROJNAME      "tpccCom50"
END

#endif // English (U.S.) resources
////////////////////////////////////

#ifndef APSTUDIO_INVOKED
////////////////////////////////////
//
// Generated from the TEXTINCLUDE 3 resource.
//
1 TYPELIB "tpccCom50.tlb"

////////////////////////////////////
#endif // not APSTUDIO_INVOKED

```

tpccCom50/tpcc_com50.rgs

```

HKCR
{
    tpccCom50.tpcc_com50.1 = s 'tpcc_com50 Class'
    {
        CLSID = s '{76F680E9-0C71-4AF3-B9F1-6B0A3AC4E1B8}'
    }
    tpccCom50.tpcc_com50 = s 'tpcc_com50 Class'
    {
        CLSID = s '{76F680E9-0C71-4AF3-B9F1-6B0A3AC4E1B8}'
        CurVer = s 'tpccCom50.tpcc_com50.1'
    }
    NoRemove CLSID
    {
        ForceRemove {76F680E9-0C71-4AF3-B9F1-6B0A3AC4E1B8}
    }
    = s 'tpcc_com50 Class'
    {
        ProgID = s 'tpccCom50.tpcc_com50.1'
        VersionIndependentProgID = s 'tpccCom50.tpcc_com50'
        InprocServer32 = s '%MODULE%'
        {
            val ThreadingModel = s 'Both'
        }
        val AppID = s '%APPID%'
        'TypeLib' = s '{7BD99942-8C52-4AEE-9B82-BECC2C7B1207}'
    }
}

```

tpccCom50/tpccCom50.rgs

```

HKCR
{
    NoRemove AppID
    {

```

```

        '%APPID%' = s 'tpccCom50'
        'tpccCom50.DLL'
    {
        val AppID = s '%APPID%'
    }
}

```

tpccCom50/tpcc_com9.cpp

// tpcc_com9.cpp : Implementation of Ctpcc_com9

```

#include "stdafx.h"
#include "tpcc_com9.h"

```

```

#include "..\tpccIsapi\tpcc.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_com9
HRESULT Ctpcc_com9::Activate()
{
    HRESULT hr = GetObjectContext(&m_spObjectContext);
    if (SUCCEEDED(hr))
    {
        DEBUGMSG("Object assigned to thread."<<endl);
        return S_OK;
    }
    return hr;
}

```

```

BOOL Ctpcc_com9::CanBePooled()
{
    DEBUGMSG("CanBePooled() returning true"<<endl);
    return TRUE;
}

```

```

void Ctpcc_com9::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_com9::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_com9::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);

    do_stok(stok,connectHandle);

    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_com9::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_com9::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_com9::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);

    do_ords(ords,connectHandle);

```

```

    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_com9::doDBInfo(void)
{
    //DUMMY FUNCTION TO ALLOW ISAPI TO CALL COM
    FUNCTION DURING LOGIN SO CONNECTIONS
    //TO DATABASE WILL BE SETUP.....I HOPE.
    DEBUGMSG("Stub function to warm object pool"<<endl);

    return S_OK;
}

/*
*****
** Name      : loadLibrary
** Description :
**          Function loads apprioate db library based on
          registry setting
** Parameters:
** Returns    :
**          int - return code
** Comments   :
**
*****
*/

Ctpcc_com9::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
{
    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,"disconnect_db")
) == NULL )
        return ERR_DISCONNECT_ADDRESS_NOT_FOUND;
    DEBUGMSG("do_disconnect
address:"<<DEBUGADDRESS(do_disconnect)<<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
address:"<<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 registry
**      dbName, dbUserName, and dbUserPassword
*****
*/

```

```

Ctpcc_com9::readRegistry()
{
    //open registry key
    HKEY registryKey;
    DWORD regType;
    char value[MAX_STRING_LEN];
    DWORD regValue;
    DWORD regValueSize = MAX_STRING_LEN;

    DEBUGMSG("Entered readRegistry(), opening key:"<<
    REGISTRY_SUB_KEY<<endl);
    //open up registry key
    if(RegOpenKeyEx(HKEY_LOCAL_MACHINE,REGISTRY_SUB_
    KEY,0,KEY_READ,&registryKey) == ERROR_SUCCESS)
    {
        DEBUGMSG(REGISTRY_SUB_KEY<<" open, getting
        database type from key"<<endl);
        regValueSize = sizeof(value);
        if
        (RegQueryValueEx(registryKey,DB_TYPE,0,&regType,(BYTE *)
        &value,&regValueSize)== ERROR_SUCCESS )
            strcpy(dbType,value);
        DEBUGMSG("Database type:"<<dbType<<" from registry
        key."<<endl);

        DEBUGMSG("Getting database name from registry
        key."<<endl);
        regValueSize = sizeof(value);
        if
        (RegQueryValueEx(registryKey,DB_NAME,0,&regType,(BYTE *)
        &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(registryKey,NULL_DB,0,&regType,(BYTE
*)&regValue,&regValueSize) == ERROR_SUCCESS)
            nullDB = regValue;
        DEBUGMSG("Null database flag:"<<nullDB<<endl);

        return OK;
    }

    DEBUGMSG("Error, unable to open registry key."<<endl);
    return ERR_UNABLE_TO_OPEN_REG;
}

/*
*****
** Name      : connectDB
** Description :
**      Function connects to the db
** Parameters:
** Returns   :
**      int - return code
** Comments  :
**
*****
*/
Ctpcc_com9::connectDB()
{
    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)
        {
            DEBUGMSG("Setting Context handle in use to true"<<endl);
            connectHandleInUse = 1;
            connected = do_connection(dbName,&connectHandle);
            if(connected != OK)
            {
                DEBUGMSG("Object do_connect failed,
rc:"<<connected<<endl);
                ERRORMSG("Object do_connect failed,
rc:"<<connected<<endl);
                return connected;
            }
            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;
}

```

tpccCom50/tpcc_com9.h

// tpcc_com9.h : Declaration of the Ctpcc_com9

```

#pragma once
#include "tpccCom9.h"
#include "resource.h"// main symbols
#include <comsvcs.h>

#include "..\tpcc\api\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 int comServerID      = 0;
static ofstream debugStream;
static ofstream errorStream;
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);

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_com9
class ATL_NO_VTABLE Ctpcc_com9 :
public CComObjectRootEx<CComMultiThreadModel>,
public IObjectControl,
public CComCoClass<Ctpcc_com9, &CLSID_tpcc_com9>,
public Itpcc_com9
{
public:
    Ctpcc_com9()
    {
        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_debug.txt
");
            }
        }
    }

```

```

        debugStream.rdbuf( )->open(comLogFile,ios_base::in |
ios_base::out | ios_base::app);

        debugFileOpen = 1;
    }
}

//open error log file
if(!errorFileOpen)
{
    InitializeCriticalSection(&errorMutex);

    char errorLogFile[128];

    sprintf(errorLogFile,"C:\\inetpub\\wwwroot\\tpcc\\comLog_err.txt");

    errorStream.rdbuf( )->open(errorLogFile,ios_base::in |
ios_base::out | ios_base::app);

    errorFileOpen=1;
}

//get registry values
if((rc = readRegistry()) != OK)
{
    ERRORMSG("Unable to open registry key " <<
REGISTRY_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_COM9)

BEGIN_COM_MAP(Ctpcc_com9)

```

```

    COM_INTERFACE_ENTRY(Itpcc_com9)
    COM_INTERFACE_ENTRY(IObjectControl)
END_COM_MAP()

// IObjectControl
public:
    STDMETHODCALLTYPE(Activate)();
    STDMETHODCALLTYPE(BOOL, CanBePooled)();
    STDMETHODCALLTYPE(void, Deactivate)();
    CComPtr<IObjectContext> m_spObjectContext;

// Itpcc_com9
public:
    STDMETHODCALLTYPE(doStockLevel)(INT *size, UCHAR **buffer);
    STDMETHODCALLTYPE(doNewOrder)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doPayment)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doOrderStatus)(INT* size, UCHAR** buffer);
    STDMETHODCALLTYPE(doDBInfo)(void);
    STDMETHODCALLTYPE(doSetComplete)(void);

    int connected;
    int connectHandleInUse;

private:
    //db2 specific context
    void *connectHandle;
    int loadLibrary();
    int readRegistry();
    int connectDB();

};

```

```

OBJECT_ENTRY_AUTO(__uuidof(tpcc_com9), Ctpcc_com9)

```

Appendix - B: Tunable Parameters

B.1 Database Parameters.

p_run.ora

```
compatible = 10.1.0.0.0
timed_statistics = FALSE
query_rewrite_enabled=false
db_name = tpcc
control_files = /home/oracle/dev//control_001
recovery_parallelism = 32
dml_locks = 500
log_buffer = 8388608
parallel_max_servers = 0
db_files = 300
fast_start_io_target = 0
db_cache_size = 18000M
db_8k_cache_size = 456M
db_16k_cache_size = 12050M
db_keep_cache_size = 75000M
db_recycle_cache_size = 6000M
shared_pool_size = 4000M
enqueue_resources = 60000
processes = 200
sessions = 200
transactions = 190
cursor_space_for_time = TRUE
db_block_size = 2048
undo_management = auto
UNDO_TABLESPACE = undo_1
trace_enabled = FALSE
db_block_checksum = FALSE
trace_enabled = FALSE
statistics_level = basic
plsql_optimize_level = 2
pga_aggregate_target = 0
undo_retention = 3
fast_start_mttr_target = 0
db_writer_processes = 2
log_checkpoint_interval = 86413230
log_checkpoints_to_alert = TRUE
log_checkpoint_timeout = 1700
java_pool_size = 0
remote_login_passwordfile = shared
disk_asynch_io = TRUE
db_block_checking = FALSE
cursor_space_for_time = TRUE
lock_sga = TRUE
hash_join_enabled = FALSE
replication_dependency_tracking = FALSE
db_file_multiblock_read_count = 1
max_dump_file_size=5M
aq_tm_processes = 0
```

B.2 Transaction Monitor Parameters

```
=====+++++ tpccCom0.tpcc_com_settings.txt
+++++=====
tpccCom0.tpcc_com_settings.txt
```

Transactions not supported
Enable object pooling
Minimum pool size 19
Maximum pool size 19
Creation timeout 180,000
Enable Object Construction

Enable Just in time activation
Concurrency Required

```
=====+++++ InetInfo_registry.reg +++++=====
InetInfo_registry.reg
```

Windows Registry Editor Version 5.00

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo\Parameters]
"ListenBackLog"=dword:000000fa
"DispatchEntries"=hex(7):4c,00,44,00,41,00,50,00,53,00,56,00,43,00,00,00,00,00
"MaxConnections"=dword:00007d00
"PoolThreadLimit"=dword:00000320
"ThreadTimeout"=dword:00015180
"MaxConcurrency"=dword:ffffff
```

Windows Registry Editor Version 5.00

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{0435C97F-9186-473F-B181-5449A2CF0042}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):39,00,2e,00,33,00,2e,00,31,00,34,00,34,00,2e,00,35,00,36,00,00,00,00,00
"SubnetMask"=hex(7):32,00,35,00,35,00,2e,00,32,00,35,00,35,00,2e,00,32,00,35,00,00,00,00,00
"DefaultGateway"=hex(7):39,00,2e,00,33,00,2e,00,31,00,34,00,34,00,2e,00,31,00,00,00,00,00
"DefaultGatewayMetric"=hex(7):31,00,00,00,00,00
"NameServer"="9.0.7.1"
"Domain"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,00,00,00,00,00,00
"TcpWindowSize"=dword:00008000
```

Windows Registry Editor Version 5.00

```
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{1E07A95A-92A0-4836-BF73-7AE38F8ACA07}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000001
"IPAddress"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00,00,00
"SubnetMask"=hex(7):30,00,2e,00,30,00,2e,00,30,00,2e,00,30,00,00,00,00,00,00,00
"DefaultGateway"=hex(7):00,00
"DefaultGatewayMetric"=hex(7):00,00
"NameServer"=""
"Domain"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
```



```

"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"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,\
0,30,00,30,00,\
34,00,00,00,00,00
"DhcpServer"="255.255.255.255"
"Lease"=dword:00000e10
"LeaseObtainedTime"=dword:40b5ddf3
"T1"=dword:40b5e4fb
"T2"=dword:40b5ea41
"LeaseTerminatesTime"=dword:40b5ec03
"IPAutoconfigurationAddress"="0.0.0.0"
"IPAutoconfigurationMask"="255.255.0.0"
"IPAutoconfigurationSeed"=dword:00000000
"AddressType"=dword:00000000
"TcpWindowSize"=dword:00040000
"DhcpClassIdBin"=hex:

```

Windows Registry Editor Version 5.00

```

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{BEABCC14-9C0A-4BE9-9817-14C4092418D3}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):31,00,30,00,2e,00,31,00,2e,00,32,00,2e,00,31,00,00,00,00,00
"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"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,\
0,30,00,30,00,\
38,00,00,00,00,00
"TcpWindowSize"=dword:00008000

```

Windows Registry Editor Version 5.00

```

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters\Interfaces\{CD3F7746-9E60-4E22-9A40-7BC6CC6B2E2E}]
"UseZeroBroadcast"=dword:00000000
"EnableDeadGWDetect"=dword:00000001
"EnableDHCP"=dword:00000000
"IPAddress"=hex(7):39,00,2e,00,33,00,2e,00,31,00,34,00,34,00,2e,00,35,00,30,00,\
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):39,00,2e,00,33,00,2e,00,31,00,34,00,34,00,2e,00,31,00,\
00,00,00,00
"DefaultGatewayMetric"=hex(7):31,00,00,00,00,00,00
"NameServer"=""
"Domain"=""
"DisableDynamicUpdate"=dword:00000000
"EnableAdapterDomainNameRegistration"=dword:00000000
"InterfaceMetric"=dword:00000001
"TCPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"UDPAllowedPorts"=hex(7):30,00,00,00,00,00,00
"RawIPAllowedProtocols"=hex(7):30,00,00,00,00,00,00
"NTEContextList"=hex(7):30,00,78,00,30,00,30,00,30,00,30,00,30,00,30,00,\
0,30,00,30,00,\
39,00,00,00,00,00
"TcpWindowSize"=dword:00008000
"DhcpIPAddress"="0.0.0.0"
"DhcpSubnetMask"="255.0.0.0"

```

```

=====+++++ Tpcc_software_registry.reg +++++=====
Tpcc_software_registry.reg

```

Windows Registry Editor Version 5.00

```

[HKEY_LOCAL_MACHINE\SOFTWARE\TPCC]
"dlvyLogPath"="c:\inetpub\wwwroot\tpcc\"
"dlvyQueueLen"=dword:00009c40
"nullDB"=dword:00000000
"htmlTrace"=dword:00000000
"dbName"="oratpcc.world"
"errorLogFile"="c:\inetpub\wwwroot\tpcc\isapi_err.log"
"htmlTraceLogFile"="c:\inetpub\wwwroot\tpcc\isapi.log"
"numUsers"=dword:00005014
"dbType"="ORACLE"
"dbUserName"="tpcc"
"dbPassword"="tpcc"
"dbInterfacePath"="C:\inetpub\wwwroot\tpcc\oracleglue.dll"
"dlvyQueueThreshold"=dword:0000000a
"dlvyThreads"=dword:00000002
"dynamicDlvy"=dword:00000001
"OraComServLog"="c:\inetpub\wwwroot\tpcc\server_print.out"
"OracleHome"="c:/Ora10g/"
"isapi_trace"=dword:00000000
"numPools"=dword:00000001
"numServers"=dword:00000001
"numWarehouse"=dword:000007f8
=====+++++ W3SVC_registry.reg +++++=====
W3SVC_registry.reg

```

Windows Registry Editor Version 5.00

```

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC]
"Type"=dword:00000020
"Start"=dword:00000002
"ErrorControl"=dword:00000001
"ImagePath"=hex(2):43,00,3a,00,5c,00,57,00,49,00,4e,00,4e,00,54,00,5c,00,53,00,\
79,00,73,00,74,00,65,00,6d,00,33,00,32,00,5c,00,69,00,6e,00,65,00,74,00,73,\
00,72,00,76,00,5c,00,69,00,6e,00,65,00,74,00,69,00,6e,00,66,00,6f,00,2e,00,\
65,00,78,00,65,00,00,00
"DisplayName"="World Wide Web Publishing Service"
"DependOnService"=hex(7):49,00,49,00,53,00,41,00,44,00,4d,00,49,00,4e,00,00,00,\
00,00
"DependOnGroup"=hex(7):00,00
"ObjectName"="LocalSystem"
"Description"="Provides Web connectivity and administration through the Internet Information Services snap-in."

```

```

"FailureActions"=hex:ff,ff,ff,ff,00,00,00,00,00,00,00,00,03,00,00,00,8
0,94,08,\

00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,00,
,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\ASP]
"NOTE"="This is for backward compatibility only."

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\ASP\Parameters]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Parameters]
"MajorVersion"=dword:00000005
"MinorVersion"=dword:00000000
"InstallPath"="C:\WINNT\System32\inetrv"
"CertMapList"="C:\WINNT\System32\inetrv\iisrmap.dll"
"AccessDeniedMessage"="Error: Access is Denied."
"Filter DLLs"=""
"LogFileDirectory"="C:\WINNT\System32\LogFiles"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Parameters\ADCLaunch]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Parameters\ADCLaunch\AdvancedDataFactory]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Parameters\ADCLaunch\RDSServer.DataFactory]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Parameters\Script Map]

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Parameters\Virtual Roots]
"/"="c:\inetpub\wwwroot,,201"
"/Scripts"="c:\inetpub\scripts,,204"
"/IISHelp"="c:\winnt\help\iishelp,,201"
"/IISAdmin"="C:\WINNT\System32\inetrv\iisadmin,,201"
"/IISamples"="c:\inetpub\iissamples,,201"
"/MSADC"="c:\program files\common files\system\msadc,,205"
"/_vti_bin"="C:\Program Files\Common Files\Microsoft
Shared\Web Server Extensions\40\isapi,,205"
"/Printers"="C:\WINNT\web\printers,,201"
"/tpcc"="C:\inetpub\wwwroot\tpcc,,207"

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Performance]
"Library"="w3ctrs.dll"
"Open"="OpenW3PerformanceData"
"Close"="CloseW3PerformanceData"
"Collect"="CollectW3PerformanceData"
"Last Counter"=dword:000008e6
"Last Help"=dword:000008e7
"First Counter"=dword:00000844
"First Help"=dword:00000845
"Library Validation
Code"=hex:4a,82,e5,79,9f,41,c4,01,10,3d,00,00,00,00,00,00
"WbemAdapFileTime"=hex:00,70,c1,b0,95,36,c3,01
"WbemAdapFileSize"=dword:00001d10
"WbemAdapStatus"=dword:00000000

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Security]
"Security"=hex:01,00,14,80,a0,00,00,ac,00,00,00,14,00,00,00,30,
00,00,00,02,\

00,1c,00,01,00,00,00,02,80,14,00,ff,01,0f,00,01,01,00,00,00,00,0
1,00,00,\

```

```

00,00,02,00,70,00,04,00,00,00,00,00,18,00,fd,01,02,00,01,01,00,00,
00,00,00,\

05,12,00,00,00,74,00,6f,00,00,00,1c,00,ff,01,0f,00,01,02,00,00,00,0
0,00,05,\

20,00,00,00,20,02,00,00,72,00,73,00,00,00,18,00,8d,01,02,00,01,01
,00,00,00,\

00,00,05,0b,00,00,00,20,02,00,00,00,00,1c,00,fd,01,02,00,01,02,00,
00,00,00,\

00,05,20,00,00,00,23,02,00,00,72,00,73,00,01,01,00,00,00,00,05
,12,00,00,\
    00,01,01,00,00,00,00,00,05,12,00,00,00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W
3SVC\Enum]
"0"="Root\LEGACY_W3SVC\0000"
"Count"=dword:00000001
"NextInstance"=dword:00000001

```

B.3 AIX Parameters

IBM eServer p5 570 Model 9117-570

OS PARAMETERS

SW_dist_intr	false	Enable SW distribution of interrupts
True		
autorestart	true	Automatically REBOOT system after a
crash	True	
boottype	disk	N/A
capacity_inc	1.00	Processor capacity increment
False		
capped	true	Partition is capped
False		
conslogin	enable	System Console Login
False		
cpuguard	enable	CPU Guard
True		
dedicated	true	Partition is dedicated
False		
ent_capacity	4.00	Entitled processor capacity
False		
frequency	400000000	System Bus Frequency
False		
fullcore	false	Enable full CORE dump
True		
fwversion	IBM,SF230_126	Firmware version and revision
levels	False	
id_to_partition	0X800003C9D0B00001	Partition ID
False		
id_to_system	0X800003C9D0B00000	System ID
False		
iostat	false	Continuously maintain DISK I/O history
True		
keylock	normal	State of system keylock at boot time
False		
max_capacity	4.00	Maximum potential processor capacity
False		
max_logname	9	Maximum login name length at boot
time	True	
maxbuf	20	Maximum number of pages in block I/O
BUFFER CACHE	True	

maxmbuf 0 Maximum Kbytes of real memory
 allowed for MBUFS True
 maxpout 0 HIGH water mark for pending write I/Os
 per file True
 maxuproc 20000 Maximum number of PROCESSES
 allowed per user True
 min_capacity 1.00 Minimum potential processor capacity
 False
 minpout 0 LOW water mark for pending write I/Os
 per file True
 modelname IBM,9117-570 Machine name
 False
 ncargs 6 ARG/ENV list size in 4K byte blocks
 True
 pre430core false Use pre-430 style CORE dump
 True
 pre520tune disable Pre-520 tuning compatibility mode
 True
 realmem 133169152 Amount of usable physical memory
 in Kbytes False
 rtasversion 1 Open Firmware RTAS version
 False
 sed_config select Stack Execution Disable (SED) Mode
 True
 systemid IBM,02102D53F Hardware system identifier
 False
 variable_weight 0 Variable processor capacity weight
 False

cpu_scale_memp = 8
 data_stagger_interval = 161
 defps = 1
 force_realias_lite = 0
 framesets = 2
 htabscale = n/a
 kernel_heap_psize = 16777216
 large_page_heap_size = 1073741824
 lgpg_regions = 7700
 lgpg_size = 16777216
 low_ps_handling = 1
 lru_file_repage = 1
 lru_poll_interval = 10
 lrubucket = 131072
 maxclient% = 80
 maxfree = 1088
 maxperm = 593929
 maxperm% = 80
 maxpin = 33284827
 maxpin% = 99
 mbuf_heap_psize = 4096
 memory_affinity = 1
 memory_frames = 33292288
 memplace_data = 2
 memplace_mapped_file = 2
 memplace_shm_anonymous = 2
 memplace_shm_named = 2
 memplace_stack = 2
 memplace_text = 2
 memplace_unmapped_file = 2
 mempools = 0
 minfree = 960
 minperm = 148482
 minperm% = 20
 nokilluid = 0
 npskill = 9344
 npsrpgmax = 74752
 npsrpgmin = 56064
 npsscrubmax = 74752
 npsscrubmin = 56064
 npswarn = 37376
 num_spec_dataseg = 0
 numpsblks = 1196032

page_steal_method = 0
 pagecoloring = n/a
 pinnable_frames = 653106
 pta_balance_threshold = n/a
 relalias_percentage = 0
 rpgclean = 0
 rpgcontrol = 2
 scrub = 0
 scrubclean = 0
 soft_min_lgpgs_vmpool = 0
 spec_dataseg_int = 512
 strict_maxclient = 1
 strict_maxperm = 0
 v_pinshm = 1
 vm_modlist_threshold = -1
 vmm_fork_policy = 1

Appendix - C: Database Setup Code

C.1 Database Creation Scripts

createuser.sh

```
#!/bin/sh

echo Creating user tpcc...
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/createuser >
junk 2>&1
if test $? -ne 0
then
    exit 1;
else
    exit 0;
fi
```

createts.sh

```
#created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatets.sh Fri Apr
15 15:53:01 PDT 2005
```

```
# Tablespace ware, ts size 40M (40960K)
# each file 40M (40960K)
# extents 37632K (37632K)
# 1 files
```

```
$tpcc_createts ware 1 1      40M 37632K unix 0      0 4 auto t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for ware failed. Exiting.
    exit 0
fi
```

```
# Tablespace cust, ts size 460200M (471244800K)
# each file 7800M (7987200K)
# extents 103582K (103582K)
# 59 files
```

```
$tpcc_createts cust 59 1      7800M 103582K unix 0      1 4 auto t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for cust failed. Exiting.
    exit 0
fi
```

```
# Tablespace dist, ts size 370M (378880K)
# each file 370M (378880K)
# extents 367104K (367104K)
# 1 files
```

```
$tpcc_createts dist 1 1      370M 367104K unix 0      60 4 auto t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for dist failed. Exiting.
    exit 0
fi
```

```
# Tablespace hist, ts size 45840M (46940160K)
# each file 7640M (7823360K)
# extents 102786K (102786K)
# 6 files
```

```
$tpcc_createts hist 6 1      7640M 102786K unix 0      61 4 auto t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for hist failed. Exiting.
    exit 0
```

```
fi
```

```
# Tablespace stok, ts size 516120M (528506880K)
# each file 7820M (8007680K)
# extents 106674K (106674K)
# 66 files
```

```
$tpcc_createts stok 66 1      7820M 106674K unix 0      67 4 auto t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for stok failed. Exiting.
    exit 0
fi
```

```
# Tablespace item, ts size 20M (20480K)
# each file 20M (20480K)
# extents 16892K (16892K)
# 1 files
```

```
$tpcc_createts item 1 1      20M 16892K unix 0      133 4 auto t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for item failed. Exiting.
    exit 0
fi
```

```
# Tablespace ordr, ts size 625400M (640409600K)
# each file 62540M (64040960K)
# extents 103280K (103280K)
# 10 files
```

```
$tpcc_createts ordr 10 1      62540M 103280K unix 0      134 4 16K
t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for ordr failed. Exiting.
    exit 0
fi
```

```
# Tablespace nord, ts size 5980M (6123520K)
# each file 5980M (6123520K)
# extents 611158K (611158K)
# 1 files
```

```
$tpcc_createts nord 1 1      5980M 611158K unix 0      144 4 auto t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for nord failed. Exiting.
    exit 0
fi
```

```
# Tablespace iware, ts size 30M (30720K)
# each file 30M (30720K)
# extents 23024K (23024K)
# 1 files
```

```
$tpcc_createts iware 1 1      30M 23024K unix 0      145 4 auto t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for iware failed. Exiting.
    exit 0
fi
```

```
# Tablespace icust1, ts size 12410M (12707840K)
# each file 12410M (12707840K)
# extents 397024K (397024K)
# 1 files
```

```
$tpcc_createts icust1 1 1      12410M 397024K unix 0      146 4 16K
t
if expr $? != 0 > /dev/null; then
    echo Creating tablespace for icust1 failed. Exiting.
    exit 0
fi
```

```
# Tablespace icust2, ts size 31080M (31825920K)
# each file 7770M (7956480K)
```

```

# extents 248524K (248524K)
# 4 files

$tpcc_createts icust2 4 1      7770M 248524K unix 0    147 4 auto
t
    if expr $? != 0 > /dev/null; then
        echo Creating tablespace for icust2 failed. Exiting.
        exit 0
    fi

# Tablespace idist, ts size 90M (92160K)
# each file 90M (92160K)
# extents 89024K (89024K)
# 1 files

$tpcc_createts idist 1 1      90M 89024K unix 0    151 4 auto t
    if expr $? != 0 > /dev/null; then
        echo Creating tablespace for idist failed. Exiting.
        exit 0
    fi

# Tablespace istok, ts size 36710M (37591040K)
# each file 36710M (37591040K)
# extents 1174368K (1174368K)
# 1 files

$tpcc_createts istok 1 1      36710M 1174368K unix 0    152 4 16K
t
    if expr $? != 0 > /dev/null; then
        echo Creating tablespace for istok failed. Exiting.
        exit 0
    fi

# Tablespace iitem, ts size 20M (20480K)
# each file 20M (20480K)
# extents 11264K (11264K)
# 1 files

$tpcc_createts iitem 1 1      20M 11264K unix 0    153 4 auto t
    if expr $? != 0 > /dev/null; then
        echo Creating tablespace for iitem failed. Exiting.
        exit 0
    fi

# Tablespace iordr2, ts size 27360M (28016640K)
# each file 6840M (7004160K)
# extents 102920K (102920K)
# 4 files

$tpcc_createts iordr2 4 1      6840M 102920K unix 0    154 4 auto
t
    if expr $? != 0 > /dev/null; then
        echo Creating tablespace for iordr2 failed. Exiting.
        exit 0
    fi

# Tablespace temp, ts size 92160M (94371840K)
# each file 7680M (7864320K)
# extents 201594K (201594K)
# 12 files

$tpcc_createts temp 12 1      7680M 201594K unix 1    158 4 auto
t
    if expr $? != 0 > /dev/null; then
        echo Creating tablespace for temp failed. Exiting.
        exit 0
    fi

```

addfile.sh

```
#!/bin/sh
```

```

# $1 = tablespace name
# $2 = filename
# $3 = size
# $4 = temporary ts (1) or not (0)
# global variable $tpcc_listfiles, does not execute sql

if [ X${ORACLE_TRACE} = XT ]
then
    set -x
fi

if expr x$tpcc_listfiles = xt > /dev/null; then
    echo $2 $3 >> $tpcc_bench/files.dat
    exit 0
fi

if expr $4 = 1 > /dev/null; then
    altersql="alter tablespace $1 add tempfile '$2' size $3 reuse;"
else
    altersql="alter tablespace $1 add datafile '$2' size $3 reuse
autoextend on;"
fi

$tpcc_sqlplus $tpcc_user_pass <<!
    spool addfile_$1.log
    set echo on
    $altersql
    set echo off
    spool off
    exit ;
!

```

addts.sh

```

#!/bin/sh
# $1 = tablespace name
# $2 = filename
# $3 = size
# $4 = uniform size
# $5 = block size
# $6 = temporary ts (1) or not (0)
# $7 = bitmapped manage (t) or not (f) or (d) for dictionary
# global variable $tpcc_listfiles, does not execute sql

if [ X${ORACLE_TRACE} = XT ]
then
    set -x
fi

if expr x$tpcc_listfiles = xt > /dev/null; then
    echo $2 $3 >> $tpcc_bench/files.dat
    exit 0
fi

if expr $5 = auto > /dev/null; then
    bssql=
else
    bssql="blocksize $5"
fi

if expr $6 = 1 > /dev/null; then
    createsql="create temporary tablespace $1 tempfile '$2' size $3
reuse extent management local uniform size $4;"
else
    if expr x$7 = xt > /dev/null; then
        createsql="create tablespace $1 datafile '$2' size $3 reuse extent
management local uniform size $4 segment space management
auto $bssql nologging ;"
    else
        if expr x$7 = xd > /dev/null; then

```

```

        createsql="create tablespace $1 datafile '$2' size $3 reuse
extent management dictionary nologging $bssql;"
    else
        createsql="create tablespace $1 datafile '$2' size $3 reuse
extent management local uniform size $4 segment space
management manual $bssql nologging ;"
    fi
fi
fi

$tpcc_sqlplus $tpcc_user_pass <<!
    spool createts_$1.log
    set echo on
    drop tablespace $1 including contents;
$createsql
    set echo off
    spool off
    exit ;
!

```

assigntemp.sh

```

#!/bin/sh

echo Assigning temporary tablespace to user tpcc...
$tpcc_sqlplus $tpcc_dba_user_pass @$tpcc_sql_dir/assigntemp >
junk 2>&1
if test $? -ne 0
then
    exit 1;
else
    exit 0;
fi

```

driver.sh

```

#!/bin/sh

. ./stepenv.sh

if expr $# \< 1 > /dev/null; then
    echo "$0 <starting stepname> <optional: only>"
    echo OR use:
    echo "$0 buildcreate - to build the database creation scripts"
    echo "$0 create - to create the database (after buildcreate)"
    echo "$0 steps - to list individual steps"
    exit 1
fi

if expr x$1 = xsteps > /dev/null; then
    echo stepnames are from creation scripts: $tpcc_create_steps
    echo
    echo or running steps: $tpcc_steps
    echo "use the 'only' option to only do that step (otherwise all steps
after will also be executed.)"
    echo " (e.g. $0 listfiles only)"
    echo "use the 'through' option to do a sequence of steps
(inclusively.)"
    echo " (e.g. $0 shutdowndb through startupdb-p_build)"
    exit 1
fi

startstep=$1
controlcmd=$2
endstep=$3

# Aliases for special steps
if test $startstep = buildcreate; then
    startstep=`echo $tpcc_create_steps | cut -d ' ' -f1`

```

```

fi

if test $startstep = create; then
    startstep=`echo $tpcc_steps | cut -d ' ' -f1`
fi

if test "x$controlcmd" = x; then
    endstep=
    # Since endstep is null it won't match any other steps, so we keep
going.
elif test "x$controlcmd" = xonly; then
    controlcmd=only
    # this is allowed
elif test "x$controlcmd" = xthrough; then
    actualstep=f
    for step in $tpcc_create_steps $tpcc_steps ; do
        if test "x$step" = "x$endstep"; then
            actualstep=t
        fi
    done
    if test $actualstep = f; then
        echo "Invalid step $endstep. Use $0 steps to show steps."
        exit 1
    fi
else
    echo "Invalid syntax. Use $0 by itself for help."
    exit 1
fi

echo Starting from step: $startstep

dostep=f
for step in $tpcc_create_steps $tpcc_steps ; do
    if expr $step = $startstep > /dev/null; then
        dostep=t
    fi

    if expr $dostep = t > /dev/null; then
        echo $step
        cd $tpcc_bench
        $tpcc_scripts/echo $step | cut -d -f1`.sh `echo $step | sed -e's/-
*/-/` | cut -d -f2- | sed -e's/-/ /g`
        lasterror=$?
        cd $tpcc_bench
        if test -n "`find $tpcc_bench/scripts -name *.log`"; then
            mv -f *.log `find $tpcc_bench/scripts -name *.log`
        $tpcc_bench/log/ 2>/dev/null
        else
            mv -f *.log $tpcc_bench/log/ 2>/dev/null
        fi

        if expr $lasterror != 0 > /dev/null; then
            if expr $lasterror != 99 > /dev/null; then
                echo Step $step failed. Stopping driver.
                exit 1
            else
                echo Step $step has completed and requested stop. Stopping
driver.
                exit 0
            fi
        fi
        if test "x$controlcmd" = xonly; then
            exit 0
        fi
        if test "x$endstep" = "x$step"; then
            echo The driver reached the last desired step. Stopping driver.
            exit 0
        fi
    fi
done

if expr $dostep = f > /dev/null; then

```

```
echo No such step: $1
fi
```

loadware.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -w > loadware.log 2>&1
```

loaddist.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -d > loaddist.log 2>&1
```

loaditem.sh

```
cd $tpcc_bench
$tpcc_load -M $tpcc_scale -i > loaditem.log 2>&1
```

loadhist.sh

```
#created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/evenload.sh Fri Apr 15
15:53:39 PDT 2005
rm -f loadhist*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 17600 -h -b 1 -e 2200 >> loadhist0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -h -b 2201 -e 4400 >> loadhist1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -h -b 4401 -e 6600 >> loadhist2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -h -b 6601 -e 8800 >> loadhist3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -h -b 8801 -e 11000 >> loadhist4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -h -b 11001 -e 13200 >> loadhist5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -h -b 13201 -e 15400 >> loadhist6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -h -b 15401 -e 17600 >> loadhist7.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadnord.sh

```
#created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/evenload.sh Fri Apr 15
15:53:39 PDT 2005
rm -f loadnord*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 17600 -n -b 1 -e 2200 >> loadnord0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -n -b 2201 -e 4400 >> loadnord1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -n -b 4401 -e 6600 >> loadnord2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -n -b 6601 -e 8800 >> loadnord3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -n -b 8801 -e 11000 >> loadnord4.log 2>&1 &
allprocs="$allprocs ${!}"
```

```
$tpcc_load -M 17600 -n -b 11001 -e 13200 >> loadnord5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -n -b 13201 -e 15400 >> loadnord6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -n -b 15401 -e 17600 >> loadnord7.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadordrordl.sh

```
#created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/evenload.sh Fri Apr 15
15:53:39 PDT 2005
rm -f loadordrordl*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 17600 -o ${tpcc_disks_location}dummy0.dat -b 1 -e
2200 >> loadordrordl0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -o ${tpcc_disks_location}dummy1.dat -b 2201
-e 4400 >> loadordrordl1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -o ${tpcc_disks_location}dummy2.dat -b 4401
-e 6600 >> loadordrordl2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -o ${tpcc_disks_location}dummy3.dat -b 6601
-e 8800 >> loadordrordl3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -o ${tpcc_disks_location}dummy4.dat -b 8801
-e 11000 >> loadordrordl4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -o ${tpcc_disks_location}dummy5.dat -b
11001 -e 13200 >> loadordrordl5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -o ${tpcc_disks_location}dummy6.dat -b
13201 -e 15400 >> loadordrordl6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -o ${tpcc_disks_location}dummy7.dat -b
15401 -e 17600 >> loadordrordl7.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`
```

loadcust.sh

```
#created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/evenload.sh Fri Apr 15
15:53:40 PDT 2005
rm -f loadcust*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 17600 -C -l 1 -m 375 >> loadcust0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -C -l 376 -m 750 >> loadcust1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -C -l 751 -m 1125 >> loadcust2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -C -l 1126 -m 1500 >> loadcust3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -C -l 1501 -m 1875 >> loadcust4.log 2>&1 &
```



```

allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -C -l 1876 -m 2250 >> loadcust5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -C -l 2251 -m 2625 >> loadcust6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -C -l 2626 -m 3000 >> loadcust7.log 2>&1 &
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`

```

loadstok.sh

```

/*created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/evenload.sh Fri Apr 15
15:53:40 PDT 2005
rm -f loadstok*.log
cd $tpcc_bench
allprocs=
$tpcc_load -M 17600 -S -j 1 -k 12500 >> loadstok0.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -S -j 12501 -k 25000 >> loadstok1.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -S -j 25001 -k 37500 >> loadstok2.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -S -j 37501 -k 50000 >> loadstok3.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -S -j 50001 -k 62500 >> loadstok4.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -S -j 62501 -k 75000 >> loadstok5.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -S -j 75001 -k 87500 >> loadstok6.log 2>&1 &
allprocs="$allprocs ${!}"
$tpcc_load -M 17600 -S -j 87501 -k 100000 >> loadstok7.log 2>&1
&
allprocs="$allprocs ${!}"
error=0
for curproc in $allprocs; do
    wait $curproc
    error=`expr $? + $error`
done
exit `expr $error != 0`

```

C.2 SQL Creation Scripts

createdb.sql

```

/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatedb.sh Fri Apr
15 15:53:18 PDT 2005 */
spool createdb.log

set echo on

shutdown abort

startup pfile=p_create.ora nomount
create database tpcc
    controlfile reuse
    maxinstances 1
    datafile
        '/home/oracle/dev//system_1' size 400M reuse
    logfile '/home/oracle/dev//log_1_1' size 32226M reuse,
        '/home/oracle/dev//log_1_2' size 32226M reuse
    sysaux datafile '/home/oracle/dev//tpccaux' size 120M reuse ;

```

```

create undo tablespace undo_1 datafile
    '/home/oracle/dev//roll1' size 8096M reuse blocksize 8K;

```

```

set echo off
exit sql.sqlcode

```

createuser.sql

```

spool createusertpcc.log;

set echo on;

create user tpcc identified by tpcc;

grant dba to tpcc;

```

```

set echo off;
spool off;

```

```
exit ;
```

assigntemp.sql

```

spool assigntemp.log;

set echo on;

alter user tpcc temporary tablespace temp_0;

set echo off;
spool off;

exit ;

```

createtable_ware.sql

```

/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:19 PDT 2005 */
set timing on
    set sqlblanklines on
    spool createtable_ware.log
    set echo on
    drop cluster warecluster including tables ;

```

```

create cluster warecluster (
    w_id number
)
single table
hashkeys 17600
hash is ( (w_id - 1) )
size 1448
    initrans 2
storage ( buffer_pool default )
tablespace ware_0;

```

```

create table ware (
    w_id number
    , w_ytd number
    , w_tax number
    , w_name varchar2(10)
    , w_street_1 varchar2(20)
    , w_street_2 varchar2(20)
    , w_city varchar2(20)
    , w_state char(2)
    , w_zip char(9)
)
cluster warecluster (
    w_id

```

```
);
set echo off
spool off
exit sql.sqlcode;
```

createtable_cust.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:21 PDT 2005 */
```

```
set timing on
set sqlblanklines on
spool createtable_cust.log
set echo on
drop cluster custcluster including tables ;
```

```
create cluster custcluster (
  c_id number
, c_d_id number
, c_w_id number
)
single table
hashkeys 528000000
hash is ( (c_id * ( 17600 * 10 ) + c_w_id * 10 + c_d_id ) )
size 180
pctfree 0 initrans 3
storage ( buffer_pool recycle ) parallel ( degree 4 )
tablespace cust_0;
```

```
create table cust (
  c_id number
, c_d_id number
, c_w_id number
, c_discount number
, c_credit char(2)
, c_last varchar2(16)
, c_first varchar2(16)
, c_credit_lim number
, c_balance number
, c_ytd_payment number
, c_payment_cnt number
, c_delivery_cnt number
, c_street_1 varchar2(20)
, c_street_2 varchar2(20)
, c_city varchar2(20)
, c_state char(2)
, c_zip char(9)
, c_phone char(16)
, c_since date
, c_middle char(2)
, c_data char(500)
)
cluster custcluster (
  c_id
, c_d_id
, c_w_id
);
set echo off
spool off
exit sql.sqlcode;
```

createtable_dist.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:25 PDT 2005 */
```

```
set timing on
set sqlblanklines on
spool createtable_dist.log
set echo on
drop cluster distcluster including tables ;
```

```
create cluster distcluster (
  d_id number
, d_w_id number
)
single table
hashkeys 176000
hash is ( ((d_w_id * 10) + d_id ) )
size 1448
initrans 4
storage ( buffer_pool default )
tablespace dist_0;
```

```
create table dist (
  d_id number
, d_w_id number
, d_ytd number
, d_next_o_id number
, d_tax number
, d_name varchar2(10)
, d_street_1 varchar2(20)
, d_street_2 varchar2(20)
, d_city varchar2(20)
, d_state char(2)
, d_zip char(9)
)
cluster distcluster (
  d_id
, d_w_id
);
set echo off
spool off
exit sql.sqlcode;
```

createtable_hist.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:28 PDT 2005 */
```

```
set timing on
set sqlblanklines on
spool createtable_hist.log
set echo on
drop table hist ;
```

```
create table hist (
  h_c_id number
, h_c_d_id number
, h_c_w_id number
, h_d_id number
, h_w_id number
, h_date date
, h_amount number
, h_data varchar2(24)
)
pctfree 5 initrans 4
storage ( buffer_pool recycle )
tablespace hist_0 ;
set echo off
spool off
exit sql.sqlcode;
```

createtable_stok.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:29 PDT 2005 */
```

```
set timing on
set sqlblanklines on
spool createtable_stok.log
set echo on
drop cluster stokcluster including tables ;
```

```

create cluster stokcluster (
  s_i_id number
, s_w_id number
)
single table
hashkeys 1760000000
hash is ( (s_i_id * 17600 + s_w_id) )
size 256
pctfree 0 initrans 2 maxtrans 2
storage ( buffer_pool keep ) parallel ( degree 4 )
tablespace stok_0;

```

```

create table stok (
  s_i_id number
, s_w_id number
, s_quantity number
, s_ytd number
, s_order_cnt number
, s_remote_cnt number
, s_data varchar2(50)
, s_dist_01 char(24)
, s_dist_02 char(24)
, s_dist_03 char(24)
, s_dist_04 char(24)
, s_dist_05 char(24)
, s_dist_06 char(24)
, s_dist_07 char(24)
, s_dist_08 char(24)
, s_dist_09 char(24)
, s_dist_10 char(24)
)

```

```

cluster stokcluster (
  s_i_id
, s_w_id
);
set echo off
spool off
exit sql.sqlcode;

```

createtable_item.sql

```

/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:33 PDT 2005 */
set timing on
set sqlblanklines on
spool createtable_item.log
set echo on
drop cluster itemcluster including tables ;

```

```

create cluster itemcluster (
  i_id number(6,0)
)
single table
hashkeys 100000
hash is ( (i_id) )
size 120
pctfree 0 initrans 3
storage ( buffer_pool keep )
tablespace item_0;

```

```

create table item (
  i_id number(6,0)
, i_name varchar2(24)
, i_price number
, i_data varchar2(50)
, i_im_id number
)
cluster itemcluster (
  i_id
);

```

```

set echo off
spool off
exit sql.sqlcode;

```

createtable_ordl.sql

```

/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:36 PDT 2005 */
set timing on
set sqlblanklines on
spool createtable_ordl.log
set echo on
create table ordl (
  ol_w_id number
, ol_d_id number
, ol_o_id number sort
, ol_number number sort
, ol_i_id number
, ol_delivery_d date
, ol_amount number
, ol_supply_w_id number
, ol_quantity number
, ol_dist_info char(24)
, constraint ordl_uk primary key (ol_w_id, ol_d_id, ol_o_id,
ol_number )) CLUSTER ordcluster_queue(ol_w_id, ol_d_id,
ol_o_id, ol_number) ;
set echo off
spool off
exit sql.sqlcode;

```

createtable_nord.sql

```

/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:38 PDT 2005 */
set timing on
set sqlblanklines on
spool createtable_nord.log
set echo on
drop cluster nordcluster_queue including tables ;

```

```

create cluster nordcluster_queue (
  no_w_id number
, no_d_id number
, no_o_id number SORT
)

```

```

hashkeys 176000
hash is ( (no_w_id - 1) * 10 + no_d_id - 1 )
size 190
tablespace nord_0;

```

```

create table nord (
  no_w_id number
, no_d_id number
, no_o_id number sort
, constraint nord_uk primary key ( no_w_id
, no_d_id
, no_o_id )
)
cluster nordcluster_queue (
  no_w_id
, no_d_id
, no_o_id
);
set echo off
spool off
exit sql.sqlcode;

```

createindex iware.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreateindex.sh Fri
Apr 15 15:53:41 PDT 2005 */
set timing on
  set sqlblanklines on
  spool createindex_iware.log ;
  set echo on ;
  drop index iware ;
  create unique index iware on ware ( w_id )
pctfree 1  initrans 3
storage ( buffer_pool default )
parallel 1
compute statistics
tablespace iware_0 ;
set echo off
spool off
exit sql.sqlcode;
```

createindex icust1.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreateindex.sh Fri
Apr 15 15:53:42 PDT 2005 */
set timing on
  set sqlblanklines on
  spool createindex_icust1.log ;
  set echo on ;
  drop index icust1 ;
  create unique index icust1 on cust ( c_w_id
, c_d_id
, c_id )
pctfree 1  initrans 3
storage ( buffer_pool default )
parallel 16
compute statistics
tablespace icust1_0 ;
set echo off
spool off
exit sql.sqlcode;
```

createindex icust2.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreateindex.sh Fri
Apr 15 15:53:43 PDT 2005 */
set timing on
  set sqlblanklines on
  spool createindex_icust2.log ;
  set echo on ;
  drop index icust2 ;
  create unique index icust2 on cust ( c_last
, c_w_id
, c_d_id
, c_first
, c_id )
pctfree 1  initrans 3
storage ( buffer_pool default )
parallel 16
compute statistics
tablespace icust2_0 ;
set echo off
spool off
exit sql.sqlcode;;
```

createindex idist.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreateindex.sh Fri
Apr 15 15:53:44 PDT 2005 */
set timing on
```

```
  set sqlblanklines on
  spool createindex_idist.log ;
  set echo on ;
  drop index idist ;
  create unique index idist on dist ( d_w_id
, d_id )
pctfree 5  initrans 3
storage ( buffer_pool default )
parallel 1
compute statistics
tablespace idist_0 ;
set echo off
spool off
exit sql.sqlcode;
```

createindex istok.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreateindex.sh Fri
Apr 15 15:53:45 PDT 2005 */
set timing on
  set sqlblanklines on
  spool createindex_istok.log ;
  set echo on ;
  drop index istok ;
  create unique index istok on stok ( s_i_id
, s_w_id )
pctfree 1  initrans 3
storage ( buffer_pool default )
parallel 16
compute statistics
tablespace istok_0 ;
set echo off
spool off
exit sql.sqlcode;
```

createindex iitem.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreateindex.sh Fri
Apr 15 15:53:46 PDT 2005 */
set timing on
  set sqlblanklines on
  spool createindex_iitem.log ;
  set echo on ;
  drop index iitem ;
  create unique index iitem on item ( i_id )
pctfree 5  initrans 4
storage ( buffer_pool default )

compute statistics
tablespace iitem_0 ;
set echo off
spool off
exit sql.sqlcode;
```

createindex iordr2.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreateindex.sh Fri
Apr 15 15:53:47 PDT 2005 */
set timing on
  set sqlblanklines on
  spool createindex_iordr2.log ;
  set echo on ;
  drop index iordr2 ;
  create unique index iordr2 on ordr ( o_c_id
, o_d_id
, o_w_id
, o_id )
pctfree 25  initrans 4
storage ( buffer_pool default )
```

```
parallel 16
compute statistics
tablespace iordr2_0 ;
set echo off
spool off
exit sql.sqlcode;
```

createtable_ordr.sql

```
/* created automatically by
/scratch/roagrawa/ibm/tpcc_17600/scripts/buildcreatetable.sh Fri
Apr 15 15:53:34 PDT 2005 */
set timing on
set sqlblanklines on
spool createtable_ordr.log
set echo on
drop cluster ordrcluster_queue including tables ;
```

```
create cluster ordrcluster_queue (
o_w_id number
,o_d_id number
,o_id number SORT
,o_number number SORT
)

hashkeys 176000
hash is ( (o_w_id - 1) * 10 + o_d_id - 1 )
size 1490
tablespace ordr_0;
```

```
create table ordr (
o_id number sort
,o_w_id number
,o_d_id number
,o_c_id number
,o_carrier_id number
,o_ol_cnt number
,o_all_local number
,o_entry_d date
, constraint ordr_uk primary key ( o_w_id
,o_d_id
,o_id )
)
cluster ordrcluster_queue (
o_w_id
,o_d_id
,o_id
);
set echo off
spool off
exit sql.sqlcode;
```

analyze.sql

```
spool analyze.log;
set echo on;
```

```
connect tpcc/tpcc
```

```
execute dbms_stats.GATHER_TABLE_STATS
(OWNNAME=>'TPCC', -
TABNAME=>'STOK', -
PARTNAME=>NULL, -
ESTIMATE_PERCENT=>1, -
BLOCK_SAMPLE=>TRUE, -
METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -
DEGREE=>10, -
GRANULARITY=>'DEFAULT', -
CASCADE=>TRUE);
```

```
execute dbms_stats.GATHER_TABLE_STATS
(OWNNAME=>'TPCC', -
TABNAME=>'CUST', -
PARTNAME=>NULL, -
ESTIMATE_PERCENT=>1, -
BLOCK_SAMPLE=>TRUE, -
METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -
DEGREE=>10, -
GRANULARITY=>'DEFAULT', -
CASCADE=>TRUE);
```

```
execute dbms_stats.GATHER_TABLE_STATS
(OWNNAME=>'TPCC', -
TABNAME=>'ORDR', -
PARTNAME=>NULL, -
ESTIMATE_PERCENT=>1, -
BLOCK_SAMPLE=>TRUE, -
METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -
DEGREE=>10, -
GRANULARITY=>'DEFAULT', -
CASCADE=>TRUE);
```

```
execute dbms_stats.GATHER_TABLE_STATS
(OWNNAME=>'TPCC', -
TABNAME=>'ORDL', -
PARTNAME=>NULL, -
ESTIMATE_PERCENT=>1, -
BLOCK_SAMPLE=>TRUE, -
METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -
DEGREE=>10, -
GRANULARITY=>'DEFAULT', -
CASCADE=>TRUE);
```

```
execute dbms_stats.GATHER_TABLE_STATS
(OWNNAME=>'TPCC', -
TABNAME=>'NORD', -
PARTNAME=>NULL, -
ESTIMATE_PERCENT=>1, -
BLOCK_SAMPLE=>TRUE, -
METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -
DEGREE=>10, -
GRANULARITY=>'DEFAULT', -
CASCADE=>TRUE);
```

```
execute
dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
TABNAME=>'HIST', -
PARTNAME=>NULL, -
ESTIMATE_PERCENT=>1, -
BLOCK_SAMPLE=>TRUE, -
METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -
DEGREE=>10, -
GRANULARITY=>'DEFAULT', -
CASCADE=>TRUE);
```

```
execute
dbms_stats.GATHER_TABLE_STATS(OWNNAME=>'TPCC', -
TABNAME=>'DIST', -
PARTNAME=>NULL, -
ESTIMATE_PERCENT=>1, -
BLOCK_SAMPLE=>TRUE, -
METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -
DEGREE=>10, -
GRANULARITY=>'DEFAULT', -
CASCADE=>TRUE);
```

```

execute
dbms_stats.GATHER_TABLE_STATS(OWNNNAME=>'TPCC', -
    TABNAME=>'ITEM', -
    PARTNAME=>NULL, -
    ESTIMATE_PERCENT=>10, -
    BLOCK_SAMPLE=>TRUE, -
    METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -

    DEGREE=>1, -
    GRANULARITY=>'DEFAULT', -
    CASCADE=>TRUE);

```

```

execute
dbms_stats.GATHER_TABLE_STATS(OWNNNAME=>'TPCC', -
    TABNAME=>'WARE', -
    PARTNAME=>NULL, -
    ESTIMATE_PERCENT=>10, -
    BLOCK_SAMPLE=>TRUE, -
    METHOD_OPT=>'FOR ALL COLUMNS

SIZE 1', -

    DEGREE=>10, -
    GRANULARITY=>'DEFAULT', -
    CASCADE=>TRUE);

```

```

set echo off;
spool off;

exit sql.sqlcode;

```

createspacestats.sql

```

@space_init
@space_get 203439 17600
@space_rpt
spool off
exit sql.sqlcode;

```

createstoredprocs.sql

```

spool createstoredprocs.log
@tkvcin.in.sql
spool off
exit sql.sqlcode;

```

C.3 Data Generation Code

tpccload.c

```

#ifdef RCSID
static char *RCSid =
    "$Header: tpccload.c 7030100.1 96/05/13 16:20:36 plai
    Generic<base> $ Copyr (c) 1993 Oracle";
#endif /* RCSID */

/*=====
=====+
|      Copyright (c) 1994  Oracle Corp, Redwood Shores, CA      |
|      OPEN SYSTEMS PERFORMANCE GROUP                          |
|                                                                  |
|      All Rights Reserved                                       |
|                                                                  |
+=====
=====+
| FILENAME
| tpccload.c

```

```

| DESCRIPTION
| Load or generate TPC-C database tables.
| Usage: tpccload -M <# of wares> [options]
| options: -A load all tables
|           -w load ware table
|           -d load dist table
|           -c load cust table (cluster around c_w_id)
|           -C load cust table (cluster around c_i_id)
|           -i load item table
|           -s load stok table (cluster around s_w_id)
|           -S load stok table (cluster around s_i_id)
|           -h load hist table
|           -n load new-order table
|           -o <oline file> load order and order-line table
|           -b <ware#> beginning ware number
|           -e <ware#> ending ware number
|           -j <item#> beginning item number (with -S)
|           -k <item#> ending item number (with -S)
|           -l <cid#> beginning cid number (with -C)
|           -m <cid#> ending cid number (with -C)
|           -g generate rows to standard output

```

```

+=====
=====*/

```

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <time.h>
#include <sys/types.h>
#include "tpcc.h"

```

```

#ifdef ORA_NT
#undef boolean
#include <process.h>
#include "dpbcore.h"
# define gettime dpbtimef
# define getcpu dpbcpu
# define lrand48() ((long)rand() <<15 | rand())
#ifdef __STDC__
# define PROTO(args) args
#else
# define PROTO(args) ()
#endif
#endif

```

```

#define DISTARR 10 /* dist insert array size */
#define CUSTARR 100 /* cust insert array size */
#define STOCARR 100 /* stok insert array size */
#define ITEMARR 100 /* item insert array size */
#define HISTARR 100 /* hist insert array size */
#define ORDEARR 100 /* order insert array size */
#define NEWOARR 100 /* new order insert array size */

```

```

#define DISTFAC 10 /* max. dist id */
#define CUSTFAC 3000 /* max. cust id */
#define STOCFAC 100000 /* max. stok id */
#define ITEMFAC 100000 /* max. item id */
#define HISTFAC 30000 /* history / warehouse */
#define ORDEFAC 3000 /* order / district */
#define NEWOFAC 900 /* new order / district */

```

```

#define C 0 /* constant in non-uniform dist. eqt. */
#define CNUM1 1 /* first constant in non-uniform dist. eqt. */
#define CNUM2 2 /* second constant in non-uniform dist. eqt. */
#define CNUM3 3 /* third constant in non-uniform dist. eqt. */

```

```

#define SEED 2 /* seed for random functions */

```

```

#define NOT_SERIALIZABLE 8177 /* ORA-08177: transaction not
serializable */
#define SNAPSHOT_TOO_OLD 1555 /* ORA-01555: snapshot too
old */
#define RECOVERERR -10
#define IRRECERR -20

#define SQLTXTW "INSERT INTO ware (w_id, w_ytd, w_tax,
w_name, w_street_1, w_street_2, w_city, w_state, w_zip) VALUES
(:w_id, 30000000, :w_tax, :w_name, :w_street_1, \
:w_street_2, :w_city, :w_state, :w_zip)"

#define SQLTXTD "INSERT INTO dist (d_id, d_w_id, d_ytd, d_tax,
d_next_o_id, d_name, d_street_1, d_street_2, d_city, d_state,
d_zip) VALUES (:d_id, :d_w_id, 30000000, :d_tax, \

3001, :d_name, :d_street_1, :d_street_2, :d_city, :d_state, :d_zip)"

#define SQLTXTCQUERY "select /*+ HASH ( cust ) */ count(*) from
cust where c_w_id = :s_c_w_id and c_d_id = :s_c_d_id and c_id
= :s_c_id"

#define SQLTXTC "INSERT INTO cust (C_ID, C_D_ID, C_W_ID,
C_FIRST, C_MIDDLE, C_LAST, 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, C_YTD_PAYMENT,
C_PAYMENT_CNT, C_DELIVERY_CNT, C_DATA) VALUES
(:c_id, :c_d_id, :c_w_id, \
:c_first, 'OE', :c_last, :c_street_1, :c_street_2, :c_city, :c_state, \
:c_zip, :c_phone, SYSDATE, :c_credit, 5000000, :c_discount, -
1000, 1000, 1, \
0, :c_data)"

#define SQLTXTH "INSERT INTO hist (h_c_id, h_c_d_id, h_c_w_id,
h_d_id, h_w_id, h_date, h_amount, h_data) VALUES
(:h_c_id, :h_c_d_id, :h_c_w_id, \
:h_d_id, :h_w_id, SYSDATE, 1000, :h_data)"

#define SQLXTXSQUERY "select /*+ HASH ( stok ) */ count(*) from
stok where s_w_id = :s_s_w_id and s_i_id = :s_s_i_id "

#define SQLXTXS "INSERT INTO stok (s_i_id, s_w_id,
s_quantity, s_dist_01, s_dist_02, s_dist_03, s_dist_04, s_dist_05,
s_dist_06, s_dist_07, s_dist_08, s_dist_09, s_dist_10, s_ytd,
s_order_cnt, s_remote_cnt, s_data) \
VALUES (:s_i_id, :s_w_id, :s_quantity, \
:s_dist_01, :s_dist_02, :s_dist_03, :s_dist_04, :s_dist_05, :s_dist_
06, \
:s_dist_07, :s_dist_08, :s_dist_09, :s_dist_10, 0, 0, 0, :s_data)" \

#define SQLTXTI "INSERT INTO item
(I_ID, I_IM_ID, I_NAME, I_PRICE, I_DATA) VALUES
(:i_id, :i_im_id, :i_name, :i_price, \
:i_data)"

#define SQLTXTO1 "INSERT INTO ordl (O_ID,
O_D_ID, O_W_ID, O_C_ID, O_ENTRY_D, O_CARRIER_ID, O_OL_C
NT, O_ALL_LOCAL) \
VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
SYSDATE, :o_carrier_id, :o_ol_cnt, 1)"

#define SQLTXTO2 "INSERT INTO ordl (O_ID,
O_D_ID, O_W_ID, O_C_ID, O_ENTRY_D, O_CARRIER_ID, O_OL_C
NT, O_ALL_LOCAL) \
VALUES (:o_id, :o_d_id, :o_w_id, :o_c_id, \
SYSDATE, 11, :o_ol_cnt, 1)"

#define SQLXTOL1 "INSERT INTO ordl (OL_O_ID, OL_D_ID,
OL_W_ID, OL_NUMBER, OL_DELIVERY_D, OL_I_ID,
OL_SUPPLY_W_ID, OL_QUANTITY, OL_AMOUNT,
OL_DIST_INFO) \
VALUES (:ol_o_id, :ol_d_id, \

```

```

:ol_w_id, :ol_number, SYSDATE, :ol_i_id, :ol_supply_w_id, 5, 0, \
:ol_dist_info)"

#define SQLXTOL2 "INSERT INTO ordl (OL_O_ID, OL_D_ID,
OL_W_ID, OL_NUMBER, OL_DELIVERY_D, OL_I_ID,
OL_SUPPLY_W_ID, OL_QUANTITY, OL_AMOUNT,
OL_DIST_INFO) \
VALUES (:ol_o_id, :ol_d_id, \
:ol_w_id, :ol_number, to_date('01-Jan-
1811'), :ol_i_id, :ol_supply_w_id, 5, :ol_amount, \
:ol_dist_info)"

#define SQLXTNO "INSERT INTO nord (no_o_id, no_d_id,
no_w_id) VALUES (:no_o_id, :no_d_id, :no_w_id)"

#define SQLXTENHA "alter session set
'_enable_hash_overflow'=true"
#define SQLXTDIHA "alter session set
'_enable_hash_overflow'=false"

static char *lastname[] = {
"BAR",
"OUGHT",
"ABLE",
"PRI",
"PRES",
"ESE",
"ANTI",
"CALLY",
"ATION",
"EING"
};

char num9[10];
char num16[17];
char str2[3];
char str24[15][25];
int randperm3000[3000];

void initperm();
void randstr();
void randdatastr();
void randnum();
void randlastname (char*, int);
int NURand();
void sysdate();

OCISvcCtx *tpcsrv;
OCIServer *tpcsrv;
OCIError *errhp;
OCISvcCtx *tpcsvc;
OCISession *tpcusr;

OCISmt *curw;
OCISmt *curd;
OCISmt *curc;
OCISmt *curcs;
OCISmt *curh;
OCISmt *curs;
OCISmt *curss;
OCISmt *cursi;
OCISmt *curo1;
OCISmt *curo2;
OCISmt *curol1;
OCISmt *curol2;
OCISmt *curno;

OCIBind *w_id_bp = (OCIBind *) 0;
OCIBind *w_name_bp = (OCIBind *) 0;
OCIBind *w_street1_bp = (OCIBind *) 0;
OCIBind *w_street2_bp = (OCIBind *) 0;
OCIBind *w_city_bp = (OCIBind *) 0;

```

```

OCIBind *w_state_bp = (OCIBind *) 0;
OCIBind *w_zip_bp = (OCIBind *) 0;
OCIBind *w_tax_bp = (OCIBind *) 0;

OCIBind *d_id_bp = (OCIBind *) 0;
OCIBind *d_w_id_bp = (OCIBind *) 0;
OCIBind *d_name_bp = (OCIBind *) 0;
OCIBind *d_street1_bp = (OCIBind *) 0;
OCIBind *d_street2_bp = (OCIBind *) 0;
OCIBind *d_city_bp = (OCIBind *) 0;
OCIBind *d_state_bp = (OCIBind *) 0;
OCIBind *d_zip_bp = (OCIBind *) 0;
OCIBind *d_tax_bp = (OCIBind *) 0;

OCIDefine *s_c_ret_bp = (OCIDefine *) 0;
OCIBind *s_c_id_bp = (OCIBind *) 0;
OCIBind *s_c_d_id_bp = (OCIBind *) 0;
OCIBind *s_c_w_id_bp = (OCIBind *) 0;

OCIBind *c_id_bp = (OCIBind *) 0;
OCIBind *c_d_id_bp = (OCIBind *) 0;
OCIBind *c_w_id_bp = (OCIBind *) 0;
OCIBind *c_first_bp = (OCIBind *) 0;
OCIBind *c_last_bp = (OCIBind *) 0;
OCIBind *c_street1_bp = (OCIBind *) 0;
OCIBind *c_street2_bp = (OCIBind *) 0;
OCIBind *c_city_bp = (OCIBind *) 0;
OCIBind *c_state_bp = (OCIBind *) 0;
OCIBind *c_zip_bp = (OCIBind *) 0;
OCIBind *c_phone_bp = (OCIBind *) 0;
OCIBind *c_discount_bp = (OCIBind *) 0;
OCIBind *c_credit_bp = (OCIBind *) 0;
OCIBind *c_data_bp = (OCIBind *) 0;

OCIBind *i_id_bp = (OCIBind *) 0;
OCIBind *i_im_id_bp = (OCIBind *) 0;
OCIBind *i_name_bp = (OCIBind *) 0;
OCIBind *i_price_bp = (OCIBind *) 0;
OCIBind *i_data_bp = (OCIBind *) 0;

OCIDefine *s_s_ret_bp = (OCIDefine *) 0;
OCIBind *s_s_i_id_bp = (OCIBind *) 0;
OCIBind *s_s_w_id_bp = (OCIBind *) 0;

OCIBind *s_i_id_bp = (OCIBind *) 0;
OCIBind *s_w_id_bp = (OCIBind *) 0;
OCIBind *s_quantity_bp = (OCIBind *) 0;
OCIBind *s_dist_01_bp = (OCIBind *) 0;
OCIBind *s_dist_02_bp = (OCIBind *) 0;
OCIBind *s_dist_03_bp = (OCIBind *) 0;
OCIBind *s_dist_04_bp = (OCIBind *) 0;
OCIBind *s_dist_05_bp = (OCIBind *) 0;
OCIBind *s_dist_06_bp = (OCIBind *) 0;
OCIBind *s_dist_07_bp = (OCIBind *) 0;
OCIBind *s_dist_08_bp = (OCIBind *) 0;
OCIBind *s_dist_09_bp = (OCIBind *) 0;
OCIBind *s_dist_10_bp = (OCIBind *) 0;
OCIBind *s_data_bp = (OCIBind *) 0;

OCIBind *h_c_id_bp = (OCIBind *) 0;
OCIBind *h_c_d_id_bp = (OCIBind *) 0;
OCIBind *h_c_w_id_bp = (OCIBind *) 0;
OCIBind *h_d_id_bp = (OCIBind *) 0;
OCIBind *h_w_id_bp = (OCIBind *) 0;
OCIBind *h_data_bp = (OCIBind *) 0;

OCIBind *ol_o_id_bp = (OCIBind *) 0;
OCIBind *ol_d_id_bp = (OCIBind *) 0;
OCIBind *ol_w_id_bp = (OCIBind *) 0;
OCIBind *ol_i_id_bp = (OCIBind *) 0;
OCIBind *ol_number_bp = (OCIBind *) 0;
OCIBind *ol_supply_w_id_bp = (OCIBind *) 0;

```

```

OCIBind *ol_dist_info_bp = (OCIBind *) 0;
OCIBind *ol_amount_bp = (OCIBind *) 0;

OCIBind *o_id_bp = (OCIBind *) 0;
OCIBind *o_d_id_bp = (OCIBind *) 0;
OCIBind *o_w_id_bp = (OCIBind *) 0;
OCIBind *o_c_id_bp = (OCIBind *) 0;
OCIBind *o_carrier_id_bp = (OCIBind *) 0;
OCIBind *o_ol_cnt_bp = (OCIBind *) 0;
OCIBind *o_ocnt_bp = (OCIBind *) 0;
OCIBind *o_olcnt_bp = (OCIBind *) 0;

OCIBind *no_o_id_bp = (OCIBind *) 0;
OCIBind *no_d_id_bp = (OCIBind *) 0;
OCIBind *no_w_id_bp = (OCIBind *) 0;

void myusage()
{
    fprintf(stderr, "\n");
    fprintf(stderr, "Usage: \tpccload -M <multiplier> [options]\n");
    fprintf(stderr, "options:\n");
    fprintf(stderr, "\t-A :tload all tables\n");
    fprintf(stderr, "\t-w :tload ware table\n");
    fprintf(stderr, "\t-d :tload dist table\n");
    fprintf(stderr, "\t-c :tload cust table (cluster around c_w_id)\n");
    fprintf(stderr, "\t-C :tload cust table (cluster around c_id)\n");
    fprintf(stderr, "\t-i :tload item table\n");
    fprintf(stderr, "\t-s :tload stok table (cluster around s_w_id)\n");
    fprintf(stderr, "\t-S :tload stok table (cluster around s_i_id)\n");
    fprintf(stderr, "\t-h :tload hist table\n");
    fprintf(stderr, "\t-n :tload new-order table\n");
    fprintf(stderr, "\t-o <oline file> :tload order and order-line table\n");
    fprintf(stderr, "\t-b <ware#> :tbeginning ware number\n");
    fprintf(stderr, "\t-e <ware#> :tending ware number\n");
    fprintf(stderr, "\t-j <item#> :tbeginning item number (with -S)\n");
    fprintf(stderr, "\t-k <item#> :tending item number (with -S)\n");
    fprintf(stderr, "\t-l <cid#> :tbeginning cid number (with -C)\n");
    fprintf(stderr, "\t-m <cid#> :tending cid number (with -C)\n");
    fprintf(stderr, "\t-g :tgenerate rows to standard output\n");
    fprintf(stderr, "\t $tpcc_bench must be set to the location of the\n");
    fprintf(stderr, "\tkit\n");
    exit(1);
}

int sqlfile(fnam, linebuf)
char *fnam;
text *linebuf;
{
    FILE *fd;
    int nulpt = 0;
    char realfile[512];

    sprintf(realfile, "%s", fnam);
    fd = fopen(realfile, "r");
    if (!fd)
    {
        return (0);
    }
    while (fgets((char *)linebuf + nulpt, SQL_BUF_SIZE, fd))
    {
        nulpt = strlen((char *)linebuf);
    }
    return(nulpt);
}

void quit()
{
    OCIERROR(errhp, OCI_SessionEnd ( tpcsvc, errhp, tpcusr,
    OCI_DEFAULT));
}

```



```

OCIERROR(errhp,OCIServerDetach ( tpcsrv, errhp,
OCI_DEFAULT));
OCIHandleFree((dvoid *)tpcusr, OCI_HTYPE_SESSION);
OCIHandleFree((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX);
OCIHandleFree((dvoid *)errhp, OCI_HTYPE_ERROR);
OCIHandleFree((dvoid *)tpcsrv, OCI_HTYPE_SERVER);
OCIHandleFree((dvoid *)tpcenv, OCI_HTYPE_ENV);
}

```

```

void main (argc, argv)

```

```

int argc;

```

```

char *argv[];

```

```

{
    char *uid="tpcc";
    char *pwd="tpcc";
    int scale=0;
    int i, j;
    int loop;
    int loopcount;
    int cid;
    int dwid;
    int cdid;
    int cwid;
    int sid;
    int swid;
    int olcnt;
    int nrows;
    int row;

```

```

    int w_id;
    char w_name[11];
    char w_street_1[21];
    char w_street_2[21];
    char w_city[21];
    char w_state[2];
    char w_zip[9];
    float w_tax;

```

```

    int d_id[10];
    int d_w_id[10];
    char d_name[10][11];
    char d_street_1[10][21];
    char d_street_2[10][21];
    char d_city[10][21];
    char d_state[10][2];
    char d_zip[10][9];
    float d_tax[10];

```

```

    int s_c_id;
    int s_c_d_id;
    int s_c_w_id;
    int s_c_count;

```

```

    int c_id[100];
    int c_d_id[100];
    int c_w_id[100];
    char c_first[100][17];
    char c_last[100][17];
    char c_street_1[100][21];
    char c_street_2[100][21];
    char c_city[100][21];
    char c_state[100][2];
    char c_zip[100][9];
    char c_phone[100][16];
    char c_credit[100][2];
    float c_discount[100];
    char c_data[100][501];

```

```

    int i_id[100];
    int i_im_id[100];
    int i_price[100];
    char i_name[100][25];

```

```

char i_data[100][51];

```

```

int s_s_count;
int s_s_i_id;
int s_s_w_id;

```

```

int s_i_id[100];
int s_w_id[100];
int s_quantity[100];
char s_dist_01[100][25];
char s_dist_02[100][25];
char s_dist_03[100][25];
char s_dist_04[100][25];
char s_dist_05[100][25];
char s_dist_06[100][25];
char s_dist_07[100][25];
char s_dist_08[100][25];
char s_dist_09[100][25];
char s_dist_10[100][25];
char s_data[100][51];

```

```

int h_w_id[100];
int h_d_id[100];
int h_c_id[100];
char h_data[100][25];

```

```

int o_id[100];
int o_d_id[100];
int o_w_id[100];
int o_c_id[100];
int o_carrier_id[100];
int o_ol_cnt[100];

```

```

int ol_o_id[1500];
int ol_d_id[1500];
int ol_w_id[1500];
int ol_number[1500];
int ol_i_id[1500];
int ol_supply_w_id[1500];
int ol_amount[1500];
char ol_dist_info[1500][24];
int o_cnt;
int ol_cnt;

```

```

ub2 ol_o_id_len[1500];
ub2 ol_d_id_len[1500];
ub2 ol_w_id_len[1500];
ub2 ol_number_len[1500];
ub2 ol_i_id_len[1500];
ub2 ol_supply_w_id_len[1500];
ub2 ol_dist_info_len[1500];
ub2 ol_amount_len[1500];

```

```

ub4 ol_o_id_clen;
ub4 ol_d_id_clen;
ub4 ol_w_id_clen;
ub4 ol_number_clen;
ub4 ol_i_id_clen;
ub4 ol_supply_w_id_clen;
ub4 ol_dist_info_clen;
ub4 ol_amount_clen;

```

```

ub2 o_id_len[100];
ub2 o_d_id_len[100];
ub2 o_w_id_len[100];
ub2 o_c_id_len[100];
ub2 o_carrier_id_len[100];
ub2 o_ol_cnt_len[100];

```

```

ub4 o_id_clen;
ub4 o_d_id_clen;
ub4 o_w_id_clen;

```

```

ub4 o_c_id_clen;
ub4 o_carrier_id_clen;
ub4 o_ol_cnt_clen;

text stmbuff[16*1024];

int no_o_id[100];
int no_d_id[100];
int no_w_id[100];

char sdate[30];

#ifdef ORA_NT
clock_t begin_time, end_time;
clock_t begin_cpu, end_cpu;

char *arg_ptr, **end_args;
#else
double begin_time, end_time;
double begin_cpu, end_cpu;
double gettime(), getcpu();

extern int getopt();
extern char *optarg;
extern int optind, opterr;
int opt;
#endif

char *argstr="M:AwdcCisShno:b:e:j:k:l:m:g";
int do_A=0;
int do_w=0;
int do_d=0;
int do_i=0;
int do_c=0;
int do_C=0;
int do_s=0;
int do_S=0;
int do_h=0;
int do_o=0;
int do_n=0;
int gen=0;
int bware=1;
int eware=0;
int bitem=1;
int eitem=0;
int bcid=1;
int ecid=0;

FILE *olfp=NULL;
char olfname[100];
char* basename;
int status;
#ifdef ORA_NT
char fname[100];
FILE *logfile;
#endif /* ORA_NT */

/*-----+
| Parse command line -- look for scale factor. |
+-----*/

if (argc == 1) {
    myusage ();
}

#ifdef ORA_NT
end_args = argv + argc;
for (++argv; argv < end_args; )
{
    arg_ptr = *argv++;

    if (*arg_ptr != '-')

```

```

{
    myusage ();
} else
{
    switch (arg_ptr[1]) {
    case '?': myusage ();
        break;
    case 'M': scale = atoi (*argv++);
        break;
    case 'A': do_A = 1;
        break;
    case 'w': do_w = 1;
        break;
    case 'd': do_d = 1;
        break;
    case 'c': do_c = 1;
        break;
    case 'C': do_C = 1;
        break;
    case 'i': do_i = 1;
        break;
    case 's': do_s = 1;
        break;
    case 'S': do_S = 1;
        break;
    case 'h': do_h = 1;
        break;
    case 'n': do_n = 1;
        break;
    case 'o': do_o = 1;
        strcpy (olfname, *argv++);
        break;
    case 'b': bware = atoi (*argv++);
        break;
    case 'e': eware = atoi (*argv++);
        break;
    case 'j': bitem = atoi (*argv++);
        break;
    case 'k': eitem = atoi (*argv++);
        break;
    case 'l': bcid = atoi (*argv++);
        break;
    case 'm': ecid = atoi (*argv++);
        break;
    case 'g': gen = 1;
        strcpy (fname, *argv++);
        break;
    case 'l': logfile=fopen(*argv++,"w");
        break;
    default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
        fprintf (stderr, "(reached default case in getopt ())\n");
        myusage ();
    }
}
}

#else

while ((opt = getopt (argc, argv, argstr)) != -1) {
    switch (opt) {
    case '?': myusage ();
        break;
    case 'M': scale = atoi (optarg);
        break;
    case 'A': do_A = 1;
        break;
    case 'w': do_w = 1;
        break;
    case 'd': do_d = 1;
        break;
    case 'c': do_c = 1;

```

```

        break;
    case 'C': do_C = 1;
        break;
    case 'i': do_i = 1;
        break;
    case 's': do_s = 1;
        break;
    case 'S': do_S = 1;
        break;
    case 'h': do_h = 1;
        break;
    case 'n': do_n = 1;
        break;
    case 'o': do_o = 1;
        strcpy (olfname, optarg);
        break;
    case 'b': bware = atoi (optarg);
        break;
    case 'e': eware = atoi (optarg);
        break;
    case 'j': bitem = atoi (optarg);
        break;
    case 'k': eitem = atoi (optarg);
        break;
    case 'l': bcid = atoi (optarg);
        break;
    case 'm': ecid = atoi (optarg);
        break;
    case 'g': gen = 1;
        break;
    default: fprintf (stderr, "THIS SHOULD NEVER HAPPEN!!!\n");
        fprintf (stderr, "(reached default case in getopt ())\n");
        myusage ();
}

# endif /* ORA_NT */

/*-----*
| Rudimentary error checking      |
/*-----*/

if (scale < 1) {
    fprintf (stderr, "Invalid scale factor: %d\n", scale);
    myusage ();
}

if (!(do_A || do_w || do_d || do_c || do_C || do_i || do_s || do_S ||
do_h || do_o ||
do_n)) {
    fprintf (stderr, "What should I load???\n");
    myusage ();
}

if (gen && (do_A || (do_w + do_d + do_c + do_C + do_i + do_s +
do_S + do_h + do_o +
do_n > 1))) {
    fprintf (stderr, "Can only generate table one at a time\n");
    myusage ();
}

if (do_S && (do_A || do_s)) {
    fprintf (stderr, "Cluster stock table around s_w_id or s_i_id?\n");
    myusage ();
}

if (do_C && (do_A || do_c)) {
    fprintf (stderr, "Cluster cust table around c_w_id or c_id?\n");
    myusage ();
}

if (eware <= 0)

```

```

    eware = scale;
    if (ecid <= 0)
        ecid = CUSTFAC;
    if (eitem <= 0)
        eitem = STOCFAC;

    if (do_C) {
        if ((bcid < 1) || (bcid > CUSTFAC)) {
            fprintf (stderr, "Invalid beginning cid number: %d\n", bcid);
            myusage ();
        }

        if ((ecid < bcid) || (ecid > CUSTFAC)) {
            fprintf (stderr, "Invalid ending cid number: %d\n", ecid);
            myusage ();
        }
    }
    if (do_S) {
        if ((bitem < 1) || (bitem > STOCFAC)) {
            fprintf (stderr, "Invalid beginning item number: %d\n", bitem);
            myusage ();
        }

        if ((eitem < bitem) || (eitem > STOCFAC)) {
            fprintf (stderr, "Invalid ending item number: %d\n", eitem);
            myusage ();
        }
    }
    if (do_o) {
        if ((basename = getenv ("tpcc_bench")) == NULL)
        {
            fprintf (stderr, "$tpcc_bench is not set");
            myusage ();
        }
    }

    if ((bware < 1) || (bware > scale)) {
        fprintf (stderr, "Invalid beginning warehouse number: %d\n",
bware);
        myusage ();
    }

    if ((eware < bware) || (eware > scale)) {
        fprintf (stderr, "Invalid ending warehouse number: %d\n",
eware);
        myusage ();
    }

    if (gen && do_o) {
        if ((olfp = fopen (olfname, "w")) == NULL) {
            fprintf (stderr, "Can't open '%s' for writing order lines\n",
olfname);
            myusage ();
        }
    }

}

/*-----+
| Prepare to insert into database.      |
+-----*/

sysdate (sdate);
if (!gen) {

    /* log on to Oracle */

    OCIInitialize(OCI_DEFAULT|OCI_OBJECT,(dvoid *)0,0,0,0);
    OCIEEnvInit(&tpcenv, OCI_DEFAULT, 0, (dvoid **)0);
    OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsrv,
OCI_HTYPE_SERVER, 0, (dvoid **)0);

```

```

OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&errhp,
OCI_HTYPE_ERROR, 0, (dvoid **)0);
OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcsvc,
OCI_HTYPE_SVCCTX, 0, (dvoid **)0);
OCIServerAttach(tpcsrv, errhp, (text *)0, OCI_DEFAULT);
OCIAttrSet((dvoid *)tpcsvc, OCI_HTYPE_SVCCTX, (dvoid
*)tpcsrv,
(ub4)0, OCI_ATTR_SERVER, errhp);
OCIHandleAlloc((dvoid *)tpcenv, (dvoid **)&tpcusr,
OCI_HTYPE_SESSION, 0, (dvoid **)0);
OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid
*)uid,
(ub4)strlen(uid), OCI_ATTR_USERNAME, errhp);
OCIAttrSet((dvoid *)tpcusr, OCI_HTYPE_SESSION, (dvoid
*)pwd, (ub4)strlen(pwd),
OCI_ATTR_PASSWORD, errhp);
OCIERROR(errhp, OCISessionBegin(tpcsvc, errhp, tpcusr,
OCI_CRED_RDBMS, OCI_DEFAULT));

OCIAttrSet(tpcsvc, OCI_HTYPE_SVCCTX, tpcusr, 0,
OCI_ATTR_SESSION, errhp);

fprintf(stderr, "\nConnected to Oracle userid '%s/%s'.\n", uid,
pwd);

/* open cursors and parse statement */
if (do_A || do_w) {
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&curw,
OCI_HTYPE_STMT, 0, (dvoid **)0));
OCIERROR(errhp, OCISStmtPrepare(curw, errhp, (text
*)SQLXTW,
strlen((char *)SQLXTW), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
}

if (do_A || do_d) {
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&curd,
OCI_HTYPE_STMT, 0, (dvoid **)0));
OCIERROR(errhp, OCISStmtPrepare(curd, errhp, (text
*)SQLXTD,
strlen((char *)SQLXTD), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
}

if (do_A || do_c || do_C) {
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&curc,
OCI_HTYPE_STMT, 0, (dvoid **)0));
OCIERROR(errhp, OCISStmtPrepare(curc, errhp, (text
*)SQLXTC,
strlen((char *)SQLXTC), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&curcs,
OCI_HTYPE_STMT, 0, (dvoid **)0));
OCIERROR(errhp, OCISStmtPrepare(curcs, errhp, (text
*)SQLXTCQUERY,
strlen((char *)SQLXTCQUERY), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
}

if (do_A || do_h) {
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&curh,
OCI_HTYPE_STMT, 0, (dvoid **)0));
OCIERROR(errhp, OCISStmtPrepare(curh, errhp, (text
*)SQLXTH,
strlen((char *)SQLXTH), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
}

if (do_A || do_s || do_S) {
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&cur,
OCI_HTYPE_STMT, 0, (dvoid **)0));

```

```

OCIERROR(errhp, OCISStmtPrepare(curs, errhp, (text
*)SQLXTS,
strlen((char *)SQLXTS), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&curss,
OCI_HTYPE_STMT, 0, (dvoid **)0));
OCIERROR(errhp, OCISStmtPrepare(curss, errhp, (text
*)SQLXTXSQUERY,
strlen((char *)SQLXTXSQUERY), (ub4)
OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT));
}

if (do_A || do_i) {
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&curi,
OCI_HTYPE_STMT, 0, (dvoid **)0));
OCIERROR(errhp, OCISStmtPrepare(curi, errhp, (text
*)SQLXTI,
strlen((char *)SQLXTI), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
}

if (do_A || do_o) {
int stat;
char fname[160];
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&cur1,
OCI_HTYPE_STMT, 0, (dvoid **)0));
DISCARD strcpy(fname, basename);
DISCARD strcat(fname, "");
DISCARD strcat(fname, "benchrun/blocks/load_ordordl.sql");
stat = sqlfile(fname, stmbuf);
if (!stat)
{
fprintf(stderr, "unable to open %s\n", fname);
quit();
exit(1);
}
OCIERROR(errhp, OCISStmtPrepare(curo1, errhp, stmbuf,
strlen((char *)stmbuf), (ub4) OCI_NTV_SYNTAX, (ub4)
OCI_DEFAULT));
}

if (do_A || do_n) {
OCIERROR(errhp, OCIHandleAlloc(tpcenv, (dvoid **)&curno,
OCI_HTYPE_STMT, 0, (dvoid **)0));
OCIERROR(errhp, OCISStmtPrepare(curno, errhp, (text
*)SQLXTNO,
strlen((char *)SQLXTNO), (ub4) OCI_NTV_SYNTAX,
(ub4) OCI_DEFAULT));
}

/* bind variables */

/* warehouse */

if (do_A || do_w) {
OCIERROR(errhp, OCIBindByName(curw, &w_id_bp, errhp,
(text *)("w_id"), strlen(("w_id")),
(ub1 *)&(w_id), sizeof(w_id), SQLT_INT, (dvoid *) 0,
(ub2 *)0, (ub2 *)0,
(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curw, &w_name_bp,
errhp, (text *)("w_name"), strlen(("w_name"),
(ub1 *)w_name, 11, SQLT_STR, (dvoid *) 0, (ub2 *)0,
(ub2 *)0,
(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curw, &w_street1_bp,
errhp, (text *)("w_street_1"),
strlen(("w_street_1"), (ub1 *)w_street_1, 21, SQLT_STR,
(dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

```

```

OCIERROR(errhp, OCIBindByName(curw, &w_street2_bp,
errhp, (text *)"w_street_2",
        strlen("w_street_2"), (ub1 *)w_street_2, 21, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curw, &w_city_bp, errhp,
(text *)"w_city",
        strlen("w_city"), (ub1 *)w_city, 21, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curw, &w_state_bp,
errhp, (text *)"w_state",
        strlen("w_state"), (ub1 *)w_state, 2, SQLT_CHR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curw, &w_zip_bp, errhp,
(text *)"w_zip",
        strlen("w_zip"), (ub1 *)w_zip, 9, SQLT_CHR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curw, &w_tax_bp, errhp,
(text *)"w_tax",
        strlen("w_tax"), (ub1 *) &w_tax, sizeof(w_tax), SQLT_FLT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

/* district */

if (do_A || do_d) {
OCIERROR(errhp, OCIBindByName(curd, &d_id_bp, errhp,
(text *)"d_id",
        strlen("d_id"), (ub1 *)d_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curd, &d_w_id_bp, errhp,
(text *)"d_w_id",
        strlen("d_w_id"), (ub1 *)d_w_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curd, &d_name_bp,
errhp, (text *)"d_name",
        strlen("d_name"), (ub1 *)d_name, 11, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curd, &d_street1_bp,
errhp, (text *)"d_street_1",
        strlen("d_street_1"), (ub1 *)d_street_1, 21, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curd, &d_street2_bp,
errhp, (text *)"d_street_2",
        strlen("d_street_2"), (ub1 *)d_street_2, 21, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curd, &d_city_bp, errhp,
(text *)"d_city",
        strlen("d_city"), (ub1 *)d_city, 21, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

```

```

OCIERROR(errhp, OCIBindByName(curd, &d_state_bp, errhp,
(text *)"d_state",
        strlen("d_state"), (ub1 *)d_state, 2, SQLT_CHR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curd, &d_zip_bp, errhp,
(text *)"d_zip",
        strlen("d_zip"), (ub1 *)d_zip, 9, SQLT_CHR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curd, &d_tax_bp, errhp,
(text *)"d_tax",
        strlen("d_tax"), (ub1 *)d_tax, sizeof(float), SQLT_FLT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

/* customer */

if (do_A || do_c || do_C) {
OCIERROR(errhp, OCIBindByName(curcs, &s_c_id_bp,
errhp, (text *)"s_c_id",
        strlen("s_c_id"), (ub1 *)&s_c_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curcs, &s_c_w_id_bp,
errhp, (text *)"s_c_w_id",
        strlen("s_c_w_id"), (ub1 *)&s_c_w_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curcs, &s_c_d_id_bp,
errhp, (text *)"s_c_d_id",
        strlen("s_c_d_id"), (ub1 *)&s_c_d_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIDefineByPos(curcs, &s_c_ret_bp, errhp, 1, &s_c_count, sizeof(int),
        SQLT_INT, \
        0, 0, 0, OCI_DEFAULT);

OCIERROR(errhp, OCIBindByName(curc, &c_id_bp, errhp,
(text *)"c_id",
        strlen("c_id"), (ub1 *)c_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_d_id_bp, errhp,
(text *)"c_d_id",
        strlen("c_d_id"), (ub1 *)c_d_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_w_id_bp, errhp,
(text *)"c_w_id",
        strlen("c_w_id"), (ub1 *)c_w_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_first_bp, errhp,
(text *)"c_first",
        strlen("c_first"), (ub1 *)c_first, 17, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

```

```

OCIERROR(errhp, OCIBindByName(curc, &c_last_bp, errhp,
(text *)"c_last",
    strlen("c_last"), (ub1 *)c_last, 17, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_street1_bp,
errhp, (text *)"c_street_1",
    strlen("c_street_1"), (ub1 *)c_street_1, 21,
    SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_street2_bp,
errhp, (text *)"c_street_2",
    strlen("c_street_2"), (ub1 *)c_street_2, 21,
    SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_city_bp, errhp,
(text *)"c_city",
    strlen("c_city"), (ub1 *)c_city, 21, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_state_bp, errhp,
(text *)"c_state",
    strlen("c_state"), (ub1 *)c_state, 2, SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_zip_bp, errhp,
(text *)"c_zip",
    strlen("c_zip"), (ub1 *)c_zip, 9, SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_phone_bp,
errhp, (text *)"c_phone",
    strlen("c_phone"), (ub1 *)c_phone, 16, SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_credit_bp,
errhp, (text *)"c_credit",
    strlen("c_credit"), (ub1 *)c_credit, 2, SQLT_CHR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_discount_bp,
errhp, (text *)"c_discount",
    strlen("c_discount"), (ub1 *)c_discount, sizeof(float),
    SQLT_FLT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curc, &c_data_bp, errhp,
(text *)"c_data",
    strlen("c_data"), (ub1 *)c_data, 501, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

/* item */

if (do_A || do_i) {
OCIERROR(errhp, OCIBindByName(curi, &i_id_bp, errhp,
(text *)"i_id",
    strlen("i_id"), (ub1 *)i_id, sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

```

```

(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curi, &i_im_id_bp, errhp,
(text *)"i_im_id",
    strlen("i_im_id"), (ub1 *)i_im_id, sizeof(int),
    SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curi, &i_name_bp, errhp,
(text *)"i_name",
    strlen("i_name"), (ub1 *)i_name, 25, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curi, &i_price_bp, errhp,
(text *)"i_price",
    strlen("i_price"), (ub1 *)i_price, sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curi, &i_data_bp, errhp,
(text *)"i_data",
    strlen("i_data"), (ub1 *)i_data, 51, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

/* stock */

if (do_A || do_s || do_S) {
OCIERROR(errhp, OCIBindByName(curss, &s_s_i_id_bp,
errhp, (text *)"s_s_i_id",
    strlen("s_s_i_id"), (ub1 *)&s_s_i_id, sizeof(int),
    SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curss, &s_s_w_id_bp,
errhp, (text *)"s_s_w_id",
    strlen("s_s_w_id"), (ub1 *)&s_s_w_id, sizeof(int),
    SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIDefineByPos(curss,&s_s_ret_bp,errhp,1,&s_s_count,sizeof(int),
    SQLT_INT,\
    0,0,0,OCI_DEFAULT);

OCIERROR(errhp, OCIBindByName(curs, &s_i_id_bp, errhp,
(text *)"s_i_id",
    strlen("s_i_id"), (ub1 *)s_i_id, sizeof(int), SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curs, &s_w_id_bp, errhp,
(text *)"s_w_id",
    strlen("s_w_id"), (ub1 *)s_w_id, sizeof(int),
    SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curs, &s_quantity_bp,
errhp, (text *)"s_quantity",
    strlen("s_quantity"), (ub1 *)s_quantity, sizeof(int),
    SQLT_INT,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

OCIERROR(errhp, OCIBindByName(curs, &s_dist_01_bp,
errhp, (text *)"s_dist_01",
    strlen("s_dist_01"), (ub1 *)s_dist_01, 25, SQLT_STR,
    (dvoid *) 0, (ub2 *)0, (ub2 *)0,
    (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

```

```

        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_02_bp,
errhp, (text *)":s_dist_02",
        strlen(":s_dist_02"), (ub1 *)s_dist_02, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_03_bp,
errhp, (text *)":s_dist_03",
        strlen(":s_dist_03"), (ub1 *)s_dist_03, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_04_bp,
errhp, (text *)":s_dist_04",
        strlen(":s_dist_04"), (ub1 *)s_dist_04, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_05_bp,
errhp, (text *)":s_dist_05",
        strlen(":s_dist_05"), (ub1 *)s_dist_05, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_06_bp,
errhp, (text *)":s_dist_06",
        strlen(":s_dist_06"), (ub1 *)s_dist_06, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_07_bp,
errhp, (text *)":s_dist_07",
        strlen(":s_dist_07"), (ub1 *)s_dist_07, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_08_bp,
errhp, (text *)":s_dist_08",
        strlen(":s_dist_08"), (ub1 *)s_dist_08, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_09_bp,
errhp, (text *)":s_dist_09",
        strlen(":s_dist_09"), (ub1 *)s_dist_09, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_dist_10_bp,
errhp, (text *)":s_dist_10",
        strlen(":s_dist_10"), (ub1 *)s_dist_10, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curs, &s_data_bp, errhp,
(text *)":s_data",
        strlen(":s_data"), (ub1 *)s_data, 51, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));
}

```

```

/* history */

if (do_A || do_h) {
    OCIERROR(errhp, OCIBindByName(curh, &h_c_id_bp, errhp,
(text *)":h_c_id",
        strlen(":h_c_id"), (ub1 *)h_c_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_c_d_id_bp,
errhp, (text *)":h_c_d_id",
        strlen(":h_c_d_id"), (ub1 *)h_d_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_c_w_id_bp,
errhp, (text *)":h_c_w_id",
        strlen(":h_c_w_id"), (ub1 *)h_w_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_d_id_bp, errhp,
(text *)":h_d_id",
        strlen(":h_d_id"), (ub1 *)h_d_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_w_id_bp, errhp,
(text *)":h_w_id",
        strlen(":h_w_id"), (ub1 *)h_w_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curh, &h_data_bp, errhp,
(text *)":h_data",
        strlen(":h_data"), (ub1 *)h_data, 25, SQLT_STR,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
        (ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

}

/* order and order_line (delivered) */

if (do_A || do_o) {
    for (i = 0; i < ORDEARR; i++) {
        o_id_len[i] = sizeof(int);
        o_d_id_len[i] = sizeof(int);
        o_w_id_len[i] = sizeof(int);
        o_c_id_len[i] = sizeof(int);
        o_carrier_id_len[i] = sizeof(int);
        o_ol_cnt_len[i] = sizeof(int);
    }

    OCIERROR(errhp, OCIBindByName(curo1, &ol_o_id_bp,
errhp, (text *)":ol_o_id",
        strlen(":ol_o_id"), (ub1 *)ol_o_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)ol_o_id_len, (ub2 *)0,
        (ub4) 15*ORDEARR, (ub4 *)&ol_o_id_clen, (ub4)
        OCI_DEFAULT));

    OCIERROR(errhp, OCIBindByName(curo1, &ol_d_id_bp,
errhp, (text *)":ol_d_id",
        strlen(":ol_d_id"), (ub1 *)ol_d_id, sizeof(int),
        SQLT_INT,
        (dvoid *) 0, (ub2 *)ol_d_id_len, (ub2 *)0,

```

```

        (ub4) 15*ORDEARR, (ub4 *) &ol_d_id_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &ol_w_id_bp,
errhp, (text *)":ol_w_id",
        strlen(":ol_w_id"), (ub1 *)ol_w_id, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)ol_w_id_len, (ub2 *)0,
(ub4) 15*ORDEARR, (ub4 *) &ol_w_id_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &ol_number_bp,
errhp, (text *)":ol_number",
        strlen(":ol_number"), (ub1 *)ol_number, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)ol_number_len, (ub2 *)0,
(ub4) 15*ORDEARR, (ub4 *) &ol_number_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &ol_i_id_bp,
errhp, (text *)":ol_i_id",
        strlen(":ol_i_id"), (ub1 *)ol_i_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)ol_i_id_len, (ub2 *)0,
(ub4) 15*ORDEARR, (ub4 *) &ol_i_id_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1,
&ol_supply_w_id_bp, errhp, (text *)":ol_supply_w_id",
        strlen(":ol_supply_w_id"), (ub1 *)ol_supply_w_id,
sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)ol_supply_w_id_len, (ub2 *)0,
(ub4) 15*ORDEARR, (ub4 *) &ol_supply_w_id_clen,
(ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &ol_dist_info_bp,
errhp, (text *)":ol_dist_info",
        strlen(":ol_dist_info"), (ub1 *)ol_dist_info, 24,
SQLT_CHR,
        (dvoid *) 0, (ub2 *)ol_dist_info_len, (ub2 *)0,
(ub4) 15*ORDEARR, (ub4 *) &ol_dist_info_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &ol_amount_bp,
errhp, (text *)":ol_amount",
        strlen(":ol_amount"), (ub1 *)ol_amount, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)ol_amount_len, (ub2 *)0,
(ub4) 15*ORDEARR, (ub4 *) &ol_amount_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_id_bp, errhp,
(text *)":o_id",
        strlen(":o_id"), (ub1 *)o_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)o_id_len, (ub2 *)0,
(ub4) ORDEARR, (ub4 *) &o_id_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_d_id_bp,
errhp, (text *)":o_d_id",
        strlen(":o_d_id"), (ub1 *)o_d_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)o_d_id_len, (ub2 *)0,
(ub4) ORDEARR, (ub4 *) &o_d_id_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_w_id_bp,
errhp, (text *)":o_w_id",
        strlen(":o_w_id"), (ub1 *)o_w_id, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)o_w_id_len, (ub2 *)0,
(ub4) ORDEARR, (ub4 *) &o_w_id_clen, (ub4)
OCI_DEFAULT));

```

```

        OCIERROR(errhp, OCIBindByName(curo1, &o_c_id_bp,
errhp, (text *)":o_c_id",
        strlen(":o_c_id"), (ub1 *)o_c_id, sizeof(int), SQLT_INT,
        (dvoid *) 0, (ub2 *)o_c_id_len, (ub2 *)0,
(ub4) ORDEARR, (ub4 *) &o_c_id_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_carrier_id_bp,
errhp, (text *)":o_carrier_id",
        strlen(":o_carrier_id"), (ub1 *)o_carrier_id, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)o_carrier_id_len, (ub2 *)0,
(ub4) ORDEARR, (ub4 *) &o_carrier_id_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_ol_cnt_bp,
errhp, (text *)":o_ol_cnt",
        strlen(":o_ol_cnt"), (ub1 *)o_ol_cnt, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)o_ol_cnt_len, (ub2 *)0,
(ub4) ORDEARR, (ub4 *) &o_ol_cnt_clen, (ub4)
OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_ocnt_bp,
errhp, (text *)":order_rows",
        strlen(":order_rows"), (ub1 *)&o_cnt, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curo1, &o_olcnt_bp,
errhp, (text *)":ordl_rows",
        strlen(":ordl_rows"), (ub1 *)&o_cnt, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    }

    /* new order */

    if (do_A || do_n) {
        OCIERROR(errhp, OCIBindByName(curno, &no_o_id_bp,
errhp, (text *)":no_o_id",
        strlen(":no_o_id"), (ub1 *)no_o_id, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curno, &no_d_id_bp,
errhp, (text *)":no_d_id",
        strlen(":no_d_id"), (ub1 *)no_d_id, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

        OCIERROR(errhp, OCIBindByName(curno, &no_w_id_bp,
errhp, (text *)":no_w_id",
        strlen(":no_w_id"), (ub1 *)no_w_id, sizeof(int),
SQLT_INT,
        (dvoid *) 0, (ub2 *)0, (ub2 *)0,
(ub4) 0, (ub4 *) 0, (ub4) OCI_DEFAULT));

    }
}

/*-----+
| Initialize random number generator          |
+-----*/

srand (SEED);
#ifdef ORA_NT
srand48 (SEED);
#endif

```



```

initperm ();

/*-----+
| Load the WAREHOUSE table.      |
+-----*/

if (do_A || do_w) {
    nrows = eware - bware + 1;

    fprintf(stderr, "Loading/generating warehouse: w%d - w%d (%d
rows)\n",
        bware, eware, nrows);

    begin_time = gettimeofday ();
    begin_cpu = getcpu ();

    for (loop = bware; loop <= eware; loop++) {

        w_tax = (float) ((lrand48 () % 2001) * 0.0001);
        randstr (w_name, 6, 10);
        randstr (w_street_1, 10, 20);
        randstr (w_street_2, 10, 20);
        randstr (w_city, 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

        if (gen) {
            printf ("%d 30000000 %6.4f %s %s %s %s %s %s\n", loop,
w_tax,
                w_name, w_street_1, w_street_2, w_city, str2, num9);
            fflush (stdout);
        }
        else {
            w_id = loop;
            strncpy (w_state, str2, 2);
            strncpy (w_zip, num9, 9);

            status = OCIStmtExecute(tpcsvc, curw, errhp, (ub4) 1, (ub4) 0,
(CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                fprintf (stderr, "Error at ware %d\n", loop);
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
        }
    }

    end_time = gettimeofday ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec.
(%10.2f cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| Load the DISTRICT table.      |
+-----*/

if (do_A || do_d) {
    nrows = (eware - bware + 1) * DISTFAC;

    fprintf (stderr, "Loading/generating district: w%d - w%d (%d
rows)\n",
        bware, eware, nrows);

    begin_time = gettimeofday ();
    begin_cpu = getcpu ();

```

```

dwid = bware - 1;

for (row = 0; row < nrows; ) {
    dwid++;

    for (i = 0; i < DISTARR; i++, row++) {
        d_tax[i] = (float) ((lrand48 () % 2001) * 0.0001);
        randstr (d_name[i], 6, 10);
        randstr (d_street_1[i], 10, 20);
        randstr (d_street_2[i], 10, 20);
        randstr (d_city[i], 10, 20);
        randstr (str2, 2, 2);
        randnum (num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';

        if (gen) {
            printf ("%d %d 3000000 %6.4f 3001 %s %s %s %s %s
%s\n",
                i + 1, dwid, d_tax[i], d_name[i], d_street_1[i],
d_street_2[i], d_city[i], str2, num9 );
        }
        else {
            d_id[i] = i + 1;
            d_w_id[i] = dwid;
            strncpy (d_state[i], str2, 2);
            strncpy (d_zip[i], num9, 9);
        }
    }

    if (gen) {
        fflush (stdout);
    }
    else {
        status = OCIStmtExecute(tpcsvc, curd, errhp, (ub4)
DISTARR, (ub4) 0,
(CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            fprintf (stderr, "Aborted at ware %d, dist 1\n", dwid);
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
    }

    end_time = gettimeofday ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec.
(%10.2f cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| Load the CUSTOMER table.      |
+-----*/

if (do_A || do_c) {

    nrows = (eware - bware + 1) * CUSTFAC * DISTFAC;

    fprintf (stderr, "Loading/generating customer: w%d - w%d (%d
rows)\n ",
        bware, eware, nrows);

    if (getenv("tpcc_hash_overflow")) {
        fprintf(stderr, "Hash overflow is enabled\n");
        OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT,
0, (dvoid**)0);
        sprintf ((char *) stmbuf, SQLTXTENHA);
    }

```

```

OCIStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
OCI_NTV_SYNTAX, OCI_DEFAULT);
OCIERROR(errhp,OCIStmtExecute(tpcsvc, curi,
errhp,1,0,0,0,OCI_DEFAULT));
OCIHandleFree(curi, OCI_HTYPE_STMT);
fprintf(stderr,"Customer loaded for horizontal partitioning\n");
}
else
{
fprintf(stderr,"Customer not loaded for horizontal
partitioning\n");
}
begin_time = gettimeofday ();
begin_cpu = gettimeofday ();

s_c_id = 1;
s_c_d_id = 1;
s_c_w_id = bware;

while (s_c_w_id <= eware) {
status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4)
0,
(CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
if (status != OCI_SUCCESS) {
OCIERROR(errhp, status);
quit ();
exit (1);
}

if (s_c_count == 0) {
s_c_w_id--;
break;
}
else s_c_w_id++;
}

if (s_c_w_id < bware ) s_c_w_id = bware;
else {
if (s_c_w_id > eware ) s_c_w_id = eware;
while (s_c_d_id <= DISTFAC) {
status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4)
0,
(CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
if (status != OCI_SUCCESS) {
fprintf(stderr, "Select failed\n");
OCIERROR(errhp, status);
quit ();
exit (1);
}
if (s_c_count == 0) {
s_c_d_id--;
break;
}
else s_c_d_id++;
}
}
if (s_c_d_id > DISTFAC) s_c_d_id = DISTFAC;

while (s_c_id <= CUSTFAC) {
status = OCIStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4)
0,
(CONST OCISnapshot*) 0, (OCISnapshot*) 0,
(ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
if (status != OCI_SUCCESS) {
OCIERROR(errhp, status);
quit ();
exit (1);
}
if (s_c_count == 0) break;
else s_c_id++;
}
}

```

```

}
if (s_c_id > CUSTFAC) {
if (s_c_d_id == DISTFAC) {
s_c_d_id=1;
s_c_w_id++;
} else {
s_c_d_id++;
}
s_c_id=1;
}

fprintf(stderr, "start at wid: %d, did: %d, cid: %d\n ",s_c_w_id,
s_c_d_id, s_c_id);
cid = s_c_id - 1;
cdid = s_c_d_id;
cwid = s_c_w_id;
nrows = (eware - s_c_w_id + 1) * DISTFAC * CUSTFAC -
(s_c_d_id - 1) * CUSTFAC - s_c_id + 1;
fprintf(stderr, "remaining rows: %d\n ", nrows);
loopcount = 0;

for (row = 0; row < nrows; ) {
for (i = 0; i < CUSTARR && row < nrows; i++, row++) {
cid++;
if (cid > CUSTFAC) { /* cycle cust id */
cid = 1; /* cheap mod */
cdid++; /* shift dist cycle */
if (cdid > DISTFAC) {
cdid = 1;
cwid++; /* shift ware cycle */
}
}
c_id[i] = cid;
c_d_id[i] = cdid;
c_w_id[i] = cwid;
if (cid <= 1000)
randlastname (c_last[i], cid - 1);
else
randlastname (c_last[i], NURand (255, 0, 999, CNUM1));
c_credit[i][1] = 'C';
if (lrnd48 () % 10)
c_credit[i][0] = 'G';
else
c_credit[i][0] = 'B';
c_discount[i] = (float)((lrnd48 () % 5001) * 0.0001);
randstr (c_first[i], 8, 16);
randstr (c_street_1[i], 10, 20);
randstr (c_street_2[i], 10, 20);
randstr (c_city[i], 10, 20);
randstr (str2, 2, 2);
randnum (num9, 9);
num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
randnum (num16, 16);
randstr (c_data[i], 300, 500);

if (gen) {
printf ("%d %d %d %s OE %s %s %s %s %s %s %s %s
%cC 5000000 %6.4f -1000 1000 1 0 %s\n",
cid, cdid, cwid, c_first[i], c_last[i],
c_street_1[i], c_street_2[i], c_city[i], str2, num9,
num16, sdate, c_credit[i][0], c_discount[i], c_data[i]);
}
else {
strncpy (c_state[i], str2, 2);
strncpy (c_zip[i], num9, 9);
strncpy (c_phone[i], num16, 16);
}
}
if (gen) {
fflush (stdout);
}
}

```

```

else {
    status = OCISStmtExecute(tpcsvc, curc, errhp, (ub4) i, (ub4)
0,
        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);

    if (status != OCI_SUCCESS) {
        fprintf(stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
            c_w_id[0], c_d_id[0], c_id[0]);
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }
}

if ((++loopcount) % 50)
    fprintf(stderr, ".");
else
    fprintf(stderr, " %d rows committed\n ", row);
}

end_time = gettimeofday ();
end_cpu = getcpu ();
fprintf(stderr, "Done. %d rows loaded/generated in %10.2f sec.
(%10.2f cpu)\n\n",
    nrows < 0 ? 0 : nrows, end_time - begin_time, end_cpu -
begin_cpu);
if (getenv("tpcc_hash_overflow")) {
    fprintf(stderr, "Hash overflow is disabled\n");
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT,
0, (dvoid**)0);
    sprintf ((char *) stmbuf, SQLTXTDIHA);
    OCISStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
        OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCISStmtExecute(tpcsvc, curi,
errhp, 1, 0, 0, 0, OCI_DEFAULT));
    OCIHandleFree(curi, OCI_HTYPE_STMT);
}
}

/*-----+
| Load the CUSTOMER table (cluster around c_id) |
+-----*/

if (do_C) {
    srand (bcid);
#ifdef ORA_NT
    srand48 (bcid);
#endif

    nrows = (ecid - bcid + 1) * (eware - bware + 1) * DISTFAC;

    fprintf (stderr, "Loading/generating customer: c%d - c%d, w%d -
w%d (%d rows)\n ",
        bcid, ecid, bware, eware, nrows);

    if (getenv("tpcc_hash_overflow")) {
        fprintf(stderr, "Hash overflow is enabled\n");
        OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT,
0, (dvoid**)0);
        sprintf ((char *) stmbuf, SQLXTXENHA);
        OCISStmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
            OCI_NTV_SYNTAX, OCI_DEFAULT);
        OCIERROR(errhp, OCISStmtExecute(tpcsvc, curi,
errhp, 1, 0, 0, 0, OCI_DEFAULT));
        OCIHandleFree(curi, OCI_HTYPE_STMT);
        fprintf (stderr, "Customer loaded for horizontal partitioning\n");
    }
    else

```

```

{
    fprintf (stderr, "Customer not loaded for horizontal
partitioning\n");
}
begin_time = gettimeofday ();
begin_cpu = getcpu ();

s_c_id = bcid;
s_c_d_id = 1;
s_c_w_id = bware;

while (s_c_id <= ecid) {
    status = OCISStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4)
0,
        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
        (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
    if (status != OCI_SUCCESS) {
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }

    if (s_c_count == 0) {
        s_c_id--;
        break;
    }
    else s_c_id++;
}

if (s_c_id < bcid ) s_c_id = bcid;
else {
    if (s_c_id > ecid ) s_c_id = ecid;
    while (s_c_w_id <= eware) {
        status = OCISStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4)
0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            fprintf (stderr, "Select failed\n");
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_c_count == 0) {
            s_c_w_id--;
            break;
        }
        else s_c_w_id++;
    }
    if (s_c_w_id > eware) s_c_w_id = eware;
    else if (s_c_w_id < bware) s_c_w_id = bware;

    while (s_c_d_id <= DISTFAC) {
        status = OCISStmtExecute(tpcsvc, curcs, errhp, (ub4) 1, (ub4)
0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_c_count == 0) break;
        else s_c_d_id++;
    }
}

if (s_c_d_id > DISTFAC) {
    s_c_d_id=1;
    if (s_c_w_id==eware) {
        s_c_w_id=bware;
        s_c_id++;
    }
}

```

```

    }
    else s_c_w_id++;
}

fprintf(stderr, "start at cid: %d, wid: %d, did: %d\n ", s_c_id,
s_c_w_id, s_c_d_id);
cid = s_c_id;
cdid = s_c_d_id - 1;
cwid = s_c_w_id;
nrows = (ecid - s_c_id + 1) * (eware - bware + 1) * DISTFAC -
(s_c_w_id - 1) * DISTFAC - s_c_d_id + 1;
fprintf(stderr, "remaining rows: %d\n ", nrows);
loopcount = 0;

for (row = 0; row < nrows; ) {
    for (i = 0; i < CUSTARR && row < nrows; i++, row++) {
        cdid++;
        if (cdid > DISTFAC) { /* cycle dist id */
            cdid = 1;
            /* cheap mod */
            cwid++;
            /* shift dist cycle */
            if (cwid > eware) {
                cwid = bware;
                /* shift ware cycle */
                cid++;
            }
        }
        c_id[i] = cid;
        c_d_id[i] = cdid;
        c_w_id[i] = cwid;
        if (cid <= 1000)
            randlastname(c_last[i], cid - 1);
        else
            randlastname(c_last[i], NURand(255, 0, 999, CNUM1));
        c_credit[i][1] = 'C';
        if (lrand48() % 10)
            c_credit[i][0] = 'G';
        else
            c_credit[i][0] = 'B';
        c_discount[i] = (float)((lrand48() % 5001) * 0.0001);
        randstr(c_first[i], 8, 16);
        randstr(c_street_1[i], 10, 20);
        randstr(c_street_2[i], 10, 20);
        randstr(c_city[i], 10, 20);
        randstr(str2, 2, 2);
        randnum(num9, 9);
        num9[4] = num9[5] = num9[6] = num9[7] = num9[8] = '1';
        randnum(num16, 16);
        randstr(c_data[i], 300, 500);

        if (gen) {
            printf("%d %d %d %s OE %s %s %s %s %s %s %s\n",
            %cC 5000000 %6.4f -1000 1000 1 0 %s\n",
            cid, cdid, cwid, c_first[i], c_last[i],
            c_street_1[i], c_street_2[i], c_city[i], str2, num9,
            num16, sdate, c_credit[i][0], c_discount[i], c_data[i]);
        }
        else {
            strncpy(c_state[i], str2, 2);
            strncpy(c_zip[i], num9, 9);
            strncpy(c_phone[i], num16, 16);
        }
    }

    if (gen) {
        fflush(stdout);
    }
    else {
        status = OCISStmtExecute(tpcsvc, curc, errhp, (ub4) i, (ub4)
0,
        (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
        (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);

```

```

        if (status != OCI_SUCCESS) {
            fprintf(stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
            c_w_id[0], c_d_id[0], c_id[0]);
            OCIERROR(errhp, status);
            quit();
            exit(1);
        }
    }

    if ((++loopcount) % 50)
        fprintf(stderr, ".");
    else
        fprintf(stderr, " %d rows committed\n ", row);
}

end_time = gettimeofday();
end_cpu = getcpu();
fprintf(stderr, "Done. %d rows loaded/generated in %10.2f sec.
(%10.2f cpu)\n\n",
        nrows < 0 ? 0 : nrows, end_time - begin_time, end_cpu -
begin_cpu);
if (getenv("tpcc_hash_overflow")) {
    fprintf(stderr, "Hash overflow is disabled\n");
    OCIHandleAlloc(tpcenv, (dvoid **)&curi, OCI_HTYPE_STMT,
0, (dvoid**)0);
    sprintf((char *) stmbuf, SQLTXTDIHA);
    OCISmtPrepare(curi, errhp, stmbuf, strlen((char *)stmbuf),
        OCI_NTV_SYNTAX, OCI_DEFAULT);
    OCIERROR(errhp, OCISmtExecute(tpcsvc, curi,
errhp, 1, 0, 0, 0, OCI_DEFAULT));
    OCIHandleFree(curi, OCI_HTYPE_STMT);
}

/*-----+
| Load the ITEM table. |
+-----*/

if (do_A || do_i) {
    nrows = ITEMFAC;

    fprintf(stderr, "Loading/generating item: (%d rows)\n ", nrows);

    begin_time = gettimeofday();
    begin_cpu = getcpu();

    loopcount = 0;

    for (row = 0; row < nrows; ) {
        for (i = 0; i < ITEMARR; i++, row++) {
            i_im_id[i] = (lrand48() % 10000) + 1;
            i_price[i] = ((lrand48() % 9901) + 100);
            randstr(i_name[i], 14, 24);
            randdatastr(i_data[i], 26, 50);

            if (gen) {
                printf("%d %d %s %d %s\n", row + 1, i_im_id[i], i_name[i],
                i_price[i], i_data[i]);
            }
            else {
                i_id[i] = row + 1;
            }
        }

        if (gen) {
            fflush(stdout);
        }
        else {
            status = OCISmtExecute(tpcsvc, curi, errhp, (ub4)
ITEMARR, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,

```

```

        (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
    if (status != OCI_SUCCESS) {
        fprintf (stderr, "Aborted at i_id %d\n", i_id[0]);
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }
}

if ((++loopcount) % 50)
    fprintf (stderr, ".");
else
    fprintf (stderr, " %d rows committed\n", row);
}

end_time = gettime ();
end_cpu = getcpu ();
fprintf (stderr, "Done. %d rows loaded/generated in %10.2f sec.
(%10.2f cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| Load the STOCK table.      |
+-----*/

if (do_A || do_s) {

    nrows = (eware - bware + 1) * STOCFAC;

    fprintf (stderr, "Loading/generating stock: w%d - w%d (%d
rows)\n",
            bware, eware, nrows);

    begin_time = gettime ();
    begin_cpu = getcpu ();

    s_s_i_id = 1;
    s_s_w_id = bware;

    while (s_s_w_id <= eware) {
        status = OCISmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4)
0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
        if (s_s_count == 0) {
            s_s_w_id--;
            break;
        }
        else s_s_w_id++;
    }

    if (s_s_w_id < bware ) s_s_w_id = bware;
    else {
        if (s_s_w_id > eware) s_s_w_id = eware;
        while (s_s_i_id <= STOCFAC) {
            status = OCISmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4)
0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
        }
    }
}

```

```

        if (s_s_count == 0) {
            break;
        }
        else s_s_i_id++;
    }
}
if (s_s_i_id > STOCFAC) {
    s_s_i_id = 1;
    s_s_w_id++;
}

fprintf(stderr, "start at s_i_id: %d, s_w_id: %d\n", s_s_i_id,
s_s_w_id);

sid = s_s_i_id - 1;
swid = s_s_w_id;
nrows = (eware - s_s_w_id + 1) * STOCFAC - (s_s_i_id - 1);
fprintf (stderr, "remaining rows: %d\n", nrows);
loopcount = 0;

for (row = 0; row < nrows; ) {
    /* added row < nrows condition on next line - alex.ni */
    for (i = 0; (i < STOCARR) && (row < nrows); i++, row++) {
        if (++sid > STOCFAC) { /* cheap mod */
            sid = 1;
            swid++;
        }
        s_quantity[i] = (lrand48 () % 91) + 10;
        randstr (s_dist_01[i], 24, 24);
        randstr (s_dist_02[i], 24, 24);
        randstr (s_dist_03[i], 24, 24);
        randstr (s_dist_04[i], 24, 24);
        randstr (s_dist_05[i], 24, 24);
        randstr (s_dist_06[i], 24, 24);
        randstr (s_dist_07[i], 24, 24);
        randstr (s_dist_08[i], 24, 24);
        randstr (s_dist_09[i], 24, 24);
        randstr (s_dist_10[i], 24, 24);
        randdatastr (s_data[i], 26, 50);

        if (gen) {
            printf ("%d %d %d %s %s %s %s %s %s %s %s %s %s\n",
0 0 %s\n",
                sid, swid, s_quantity[i], s_dist_01[i], s_dist_02[i],
                s_dist_03[i], s_dist_04[i], s_dist_05[i], s_dist_06[i],
                s_dist_07[i], s_dist_08[i], s_dist_09[i], s_dist_10[i],
                s_data[i]);
        }
        else {
            s_i_id[i] = sid;
            s_w_id[i] = swid;
        }
    }

    if (gen) {
        fflush (stdout);
    }
    else {
        /* Changed to STOCKARR to i - alex.ni */
        status = OCISmtExecute(tpcsvc, curs, errhp, (ub4) i, (ub4)
0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            fprintf (stderr, "Aborted at w_id %d, s_i_id %d\n", s_w_id[0],
s_i_id[0]);
            OCIERROR(errhp, status);
            quit ();
            exit (1);
        }
    }
}
}

```

```

        if ((++loopcount) % 50)
            fprintf(stderr, ".");
        else
            fprintf(stderr, " %d rows committed\n", row);
    }

    end_time = gettimeofday();
    end_cpu = getcpu();
    fprintf(stderr, "Done. %d rows loaded/generated in %10.2f sec. (%10.2f cpu)\n\n",
        nrows < 0 ? 0 : nrows, end_time - begin_time, end_cpu -
        begin_cpu);
}

/*-----+
| Load the STOCK table (cluster around s_i_id). |
+-----*/

if (do_S) {

    nrows = (eitem - bitem + 1) * (eware - bware + 1);

    fprintf(stderr, "Loading/generating stock: i%d - i%d, w%d - w%d (%d rows)\n",
        bitem, eitem, bware, eware, nrows);

    begin_time = gettimeofday();
    begin_cpu = getcpu();

    s_s_i_id = bitem;
    s_s_w_id = bware;

    while (s_s_i_id <= eitem) {
        status = OCISStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4)
0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {
            OCIERROR(errhp, status);
            quit();
            exit(1);
        }
        if (s_s_count == 0) {
            s_s_i_id--;
            break;
        }
        else s_s_i_id++;
    }

    if (s_s_i_id < bitem) s_s_i_id = bitem;
    else {
        if (s_s_i_id > eitem) s_s_i_id = eitem;
        while (s_s_w_id <= eware) {
            status = OCISStmtExecute(tpcsvc, curss, errhp, (ub4) 1, (ub4)
0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT | OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                OCIERROR(errhp, status);
                quit();
                exit(1);
            }
            if (s_s_count == 0) {
                break;
            }
            else s_s_w_id++;
        }
    }
    if (s_s_w_id > eware) {
        s_s_w_id = bware;
        s_s_i_id++;
    }
}

```

```

    }

    fprintf(stderr, "start at s_i_id: %d, s_w_id: %d\n", s_s_i_id,
s_s_w_id);

    sid = s_s_i_id;
    swid = s_s_w_id - 1;
    nrows = (eitem - s_s_i_id + 1) * (eware - bware + 1) - (s_s_w_id
- bware);
    fprintf(stderr, "remaining rows: %d\n", nrows);
    loopcount = 0;

    for (row = 0; row < nrows; ) {
        for (i = 0; i < STOCARR && row < nrows; i++, row++) {
            if (++swid > eware) { /* cheap mod */
                swid = bware;
                sid++;
            }
            s_quantity[i] = (lrand48() % 91) + 10;
            randstr(s_dist_01[i], 24, 24);
            randstr(s_dist_02[i], 24, 24);
            randstr(s_dist_03[i], 24, 24);
            randstr(s_dist_04[i], 24, 24);
            randstr(s_dist_05[i], 24, 24);
            randstr(s_dist_06[i], 24, 24);
            randstr(s_dist_07[i], 24, 24);
            randstr(s_dist_08[i], 24, 24);
            randstr(s_dist_09[i], 24, 24);
            randstr(s_dist_10[i], 24, 24);
            randdatastr(s_data[i], 26, 50);

            if (gen) {
                printf("%d %d %d %s %s %s %s %s %s %s %s %s\n",
0 0 %s\n",
                    sid, swid, s_quantity[i], s_dist_01[i], s_dist_02[i],
                    s_dist_03[i], s_dist_04[i], s_dist_05[i], s_dist_06[i],
                    s_dist_07[i], s_dist_08[i], s_dist_09[i], s_dist_10[i],
                    s_data[i]);
            }
            else {
                s_i_id[i] = sid;
                s_w_id[i] = swid;
            }
        }

        if (gen) {
            fflush(stdout);
        }
        else {
            status = OCISStmtExecute(tpcsvc, curs, errhp, (ub4) i, (ub4)
0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                fprintf(stderr, "Aborted at w_id %d, s_i_id %d\n",
s_w_id[0], s_i_id[0]);
                OCIERROR(errhp, status);
                quit();
                exit(1);
            }
        }

        if ((++loopcount) % 50)
            fprintf(stderr, ".");
        else
            fprintf(stderr, " %d rows committed\n", row);
    }

    end_time = gettimeofday();
    end_cpu = getcpu();
}

```

```

    fprintf(stderr, "Done. %d rows loaded/generated in %10.2f sec.
(%10.2f cpu)\n\n",
        nrows < 0 ? 0 : nrows, end_time - begin_time, end_cpu -
begin_cpu);
}

/*-----+
| Load the HISTORY table.          |
+-----*/

if (do_A || do_h) {
    nrows = (eware - bware + 1) * HISTFAC;

    fprintf(stderr, "Loading/generating history: w%d - w%d (%d
rows)\n ",
        bware, eware, nrows);

    begin_time = gettimeofday ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
        for (i = 0; i < HISTARR; i++, row++) {
            cid++;
            if (cid > CUSTFAC) { /* cycle cust id */
                cid = 1; /* cheap mod */
                cdid++; /* shift district cycle */
                if (cdid > DISTFAC) {
                    cdid = 1;
                    cwid++; /* shift warehouse cycle */
                }
            }
            h_c_id[i] = cid;
            h_d_id[i] = cdid;
            h_w_id[i] = cwid;
            randstr (h_data[i], 12, 24);
            if (gen) {
                printf ("%d %d %d %d %d %s 1000 %s\n", cid, cdid, cwid,
cdid,
                    cwid, sdate, h_data[i]);
            }
        }

        if (gen) {
            fflush (stdout);
        }
        else {
            status = OCISmtExecute(tpcsvc, curh, errhp, (ub4)
HISTARR, (ub4) 0,
                (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
                (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
            if (status != OCI_SUCCESS) {
                fprintf (stderr, "Aborted at w_id %d, d_id %d, c_id %d\n",
                    h_w_id[0], h_d_id[0], h_c_id[0]);
                OCIERROR(errhp, status);
                quit ();
                exit (1);
            }
        }

        if ((++loopcount) % 50)
            fprintf (stderr, ".");
        else
            fprintf (stderr, " %d rows committed\n ", row);
    }

    end_time = gettimeofday ();

```

```

    end_cpu = getcpu ();
    fprintf(stderr, "Done. %d rows loaded/generated in %10.2f sec.
(%10.2f cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| Load the ORDERS and ORDER-LINE table.      |
+-----*/

if (do_A || do_o) {

    int batch_olcnt;

    nrows = (eware - bware + 1) * ORDEFAC * DISTFAC;

    fprintf (stderr, "Loading/generating orders and order-line: w%d -
w%d (%d ord, ~%d ordl)\n ",
        bware, eware, nrows, nrows * 10);

    begin_time = gettimeofday ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {

        batch_olcnt = 0;

        for (i = 0; i < ORDEARR; i++, row++) {
            cid++;
            if (cid > ORDEFAC) { /* cycle cust id */
                cid = 1; /* cheap mod */
                cdid++; /* shift district cycle */
                if (cdid > DISTFAC) {
                    cdid = 1;
                    cwid++; /* shift warehouse cycle */
                }
            }
            o_carrier_id[i] = lrand48 () % 10 + 1;
            o_ol_cnt[i] = olcnt = lrand48 () % 11 + 5;

            if (gen) {
                if (cid < 2101) {
                    printf ("%d %d %d %d %s %d %d 1\n", cid, cdid, cwid,
                        randperm3000[cid - 1], sdate, o_carrier_id[i],
                        o_ol_cnt[i]);
                }
                else {
                    /* set carrierid to 11 instead of null */
                    printf ("%d %d %d %d %s 11 %d 1\n", cid, cdid, cwid,
                        randperm3000[cid - 1], sdate, o_ol_cnt[i]);
                }
            }
            else {
                o_id[i] = cid;
                o_d_id[i] = cdid;
                o_w_id[i] = cwid;
                o_c_id[i] = randperm3000[cid - 1];
                if (cid >= 2101) {
                    o_carrier_id[i] = 11;
                }
            }
        }

        for (j = 0; j < o_ol_cnt[i]; j++, batch_olcnt++) {
            ol_i_id[batch_olcnt] = sid = lrand48 () % 100000 + 1;
            if (cid < 2101)
                ol_amount[batch_olcnt] = 0;
            else

```

```

        ol_amount[batch_olcnt] = (lrand48 () % 999999 + 1) ;
        randstr (str24[j], 24, 24);

        if (gen) {
            if (cid < 2101) {
                fprintf (olfp, "%d %d %d %d %s %d %d 5 %ld %s\n",
                    cid,
                        cdid, cwid, j + 1, sdate, ol_i_id[batch_olcnt], cwid,
                        ol_amount[batch_olcnt], str24[j]);
            }
            else {
                /* Insert a default date instead of null date */
                fprintf (olfp, "%d %d %d %d 01-Jan-1811 %d %d 5 %ld
%s\n", cid,
                    cdid, cwid, j + 1, ol_i_id[batch_olcnt], cwid,
                    ol_amount[batch_olcnt], str24[j]);
            }
        }
        else {
            ol_o_id[batch_olcnt] = cid;
            ol_d_id[batch_olcnt] = cdid;
            ol_w_id[batch_olcnt] = cwid;
            ol_number[batch_olcnt] = j + 1;
            ol_supply_w_id[batch_olcnt] = cwid;
            strncpy (ol_dist_info[batch_olcnt], str24[j], 24);
        }
    }
    if (gen) {
        fflush (olfp);
    }
}

o_cnt = ORDEARR;
ol_cnt = batch_olcnt;

for (j = 0; j < batch_olcnt; j++) {
    ol_o_id_len[j] = sizeof(int);
    ol_d_id_len[j] = sizeof(int);
    ol_w_id_len[j] = sizeof(int);
    ol_number_len[j] = sizeof(int);
    ol_i_id_len[j] = sizeof(int);
    ol_supply_w_id_len[j] = sizeof(int);
    ol_dist_info_len[j] = 24;
    ol_amount_len[j] = sizeof(int);
}

for (j = batch_olcnt; j < 15*ORDEARR; j++) {
    ol_o_id_len[j] = 0;
    ol_d_id_len[j] = 0;
    ol_w_id_len[j] = 0;
    ol_number_len[j] = 0;
    ol_i_id_len[j] = 0;
    ol_supply_w_id_len[j] = 0;
    ol_dist_info_len[j] = 0;
    ol_amount_len[j] = 0;
}

o_id_clen = ORDEARR;
o_d_id_clen = ORDEARR;
o_w_id_clen = ORDEARR;
o_c_id_clen = ORDEARR;
o_carrier_id_clen = ORDEARR;
o_ol_cnt_clen = ORDEARR;

ol_o_id_clen = batch_olcnt;
ol_d_id_clen = batch_olcnt;
ol_w_id_clen = batch_olcnt;
ol_number_clen = batch_olcnt;
ol_i_id_clen = batch_olcnt;
ol_supply_w_id_clen = batch_olcnt;
ol_dist_info_clen = batch_olcnt;
ol_amount_clen = batch_olcnt;

```

```

        OCIERROR(errhp, OCIStmtExecute(tpcsvc, curo1, errhp,
(ub4) 1, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*) 0,
            (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS));

        if ((++loopcount) % 50) {
            fprintf (stderr, ".");
        }
        else {
            fprintf (stderr, " %d orders committed\n ", row);
        }
    }

    end_time = gettimeofday ();
    end_cpu = getcpu ();
    fprintf (stderr, "Done. %d orders loaded/generated in %10.2f
sec. (%10.2f cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| Load the NEW-ORDER table.          |
+-----*/

if (do_A || do_n) {
    nrows = (eware - bware + 1) * NEWOFAC * DISTFAC;

    fprintf (stderr, "Loading/generating new-order: w%d - w%d (%d
rows)\n ",
        bware, aware, nrows);

    begin_time = gettimeofday ();
    begin_cpu = getcpu ();

    cid = 0;
    cdid = 1;
    cwid = bware;
    loopcount = 0;

    for (row = 0; row < nrows; ) {
        for (i = 0; i < NEWOARR; i++, row++) {
            cid++;
            if (cid > NEWOFAC) {
                cid = 1;
                cdid++;
            }
            if (cdid > DISTFAC) {
                cdid = 1;
                cwid++;
            }
        }

        if (gen) {
            printf ("%d %d %d\n", cid + 2100, cdid, cwid);
        }
        else {
            no_o_id[i] = cid + 2100;
            no_d_id[i] = cdid;
            no_w_id[i] = cwid;
        }
    }

    if (gen) {
        fflush (stdout);
    }
    else {
        status = OCIStmtExecute(tpcsvc, curno, errhp, (ub4)
NEWOARR, (ub4) 0,
            (CONST OCISnapshot*) 0, (OCISnapshot*)
0,
            (ub4) OCI_DEFAULT |
OCI_COMMIT_ON_SUCCESS);
        if (status != OCI_SUCCESS) {

```



```

        fprintf(stderr, "Aborted at w_id %d, d_id %d, o_id %d\n", cwid,
cdid, cid + 2100);
        OCIERROR(errhp, status);
        quit ();
        exit (1);
    }
}

if ((++loopcount) % 45)
    fprintf(stderr, ".");
else
    fprintf(stderr, " %d rows committed\n ", row);
}

end_time = gettimeofday ();
end_cpu = getcpu ();
fprintf(stderr, "Done. %d rows loaded/generated in %10.2f sec.
(%10.2f cpu)\n\n",
        nrows, end_time - begin_time, end_cpu - begin_cpu);
}

/*-----+
| clean up and exit.      |
+-----*/

if (olfp)
    fclose (olfp);
if (!lgen)
    quit ();
exit (0);
}

void initperm ()
{
    int i;
    int pos;
    int temp;

    /* init randperm3000 */

    for (i = 0; i < 3000; i++)
        randperm3000[i] = i + 1;
    for (i = 3000; i > 0; i--) {
        pos = lrand48 () % i;
        temp = randperm3000[i - 1];
        randperm3000[i - 1] = randperm3000[pos];
        randperm3000[pos] = temp;
    }
}

void randstr (str, x, y)
char *str;
int x;
int y;
{
    int i, j;
    int len;

    len = (lrand48 () % (y - x + 1)) + x;
    for (i = 0; i < len; i++) {
        j = lrand48 () % 62;
        if (j < 26)
            str[i] = (char) (j + 'a');
        else if (j < 52)
            str[i] = (char) (j - 26 + 'A');
        else
            str[i] = (char) (j - 52 + '0');
    }
    str[len] = '\0';
}

```

```

void randdatastr (str, x, y)
char *str;
int x;
int y;
{
    int i, j;
    int len;
    int pos;

    len = (lrand48 () % (y - x + 1)) + x;
    for (i = 0; i < len; i++) {
        j = lrand48 () % 62;
        if (j < 26)
            str[i] = (char) (j + 'a');
        else if (j < 52)
            str[i] = (char) (j - 26 + 'A');
        else
            str[i] = (char) (j - 52 + '0');
    }
    str[len] = '\0';
    if ((lrand48 () % 10) == 0) {
        pos = (lrand48 () % (len - 8));
        str[pos] = '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';
    }
}

void randnum (str, len)
char *str;
int len;
{
    int i;

    for (i = 0; i < len; i++)
        str[i] = (char) (lrand48 () % 10 + '0');
    str[len] = '\0';
}

void randlastname (str, id)
char *str;
int id;
{
    id = id % 1000;
    strcpy (str, lastname[id / 100]);
    strcat (str, lastname[(id / 10) % 10]);
    strcat (str, lastname[id % 10]);
}

int NURand (A, x, y, cnum)
int A, x, y, cnum;
{
    int a, b;

    a = lrand48 () % (A + 1);
    b = (lrand48 () % (y - x + 1)) + x;
    return (((a | b) + cnum) % (y - x + 1)) + x;
}

void sysdate (sdate)
char *sdate;
{
    time_t tp;
    struct tm *tmptr;

```

```

time (&tp);
tmpr = localtime (&tp);
strftime (sdate, 29, "%d-%b-%Y", tmpr);
}

int ocierror(fname, lineno, errhp, status)
char *fname;
int lineno;
OCLError *errhp;
sword status;
{
    text errbuf[512];
    sb4 errcode;
    sb4 lstat;
    ub4 recno=2;

    switch (status) {
    case OCI_SUCCESS:
        break;
    case OCI_SUCCESS_WITH_INFO:
        fprintf(stderr,"Module %s Line %d\n", fname, lineno);
        fprintf(stderr,"Error - OCI_SUCCESS_WITH_INFO\n");
        lstat = OCLErrorGet (errhp, recno++, (text *) NULL, &errcode,
errbuf,
            (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
        fprintf(stderr,"Error - %s\n", errbuf);
        break;
    case OCI_NEED_DATA:
        fprintf(stderr,"Module %s Line %d\n", fname, lineno);
        fprintf(stderr,"Error - OCI_NEED_DATA\n");
        return (IRRECERR);
    case OCI_NO_DATA:
        fprintf(stderr,"Module %s Line %d\n", fname, lineno);
        fprintf(stderr,"Error - OCI_NO_DATA\n");
        return (IRRECERR);
    case OCI_ERROR:
        lstat = OCLErrorGet (errhp, (ub4) 1,
            (text *) NULL, &errcode, errbuf,
            (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
        if (errcode == NOT_SERIALIZABLE) return (errcode);
        if (errcode == SNAPSHOT_TOO_OLD) return (errcode);
        while (lstat != OCI_NO_DATA)
        {
            fprintf(stderr,"Module %s Line %d\n", fname, lineno);
            fprintf(stderr,"Error - %s\n", errbuf);
            lstat = OCLErrorGet (errhp, recno++, (text *) NULL, &errcode,
errbuf,
                (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
        }
        return (errcode);
    case OCI_INVALID_HANDLE:
        fprintf(stderr,"Module %s Line %d\n", fname, lineno);
        fprintf(stderr,"Error - OCI_INVALID_HANDLE\n");
        exit(-1);
    case OCI_STILL_EXECUTING:
        fprintf(stderr,"Module %s Line %d\n", fname, lineno);
        fprintf(stderr,"Error - OCI_STILL_EXECUTE\n");
        return (IRRECERR);
    case OCI_CONTINUE:
        fprintf(stderr,"Module %s Line %d\n", fname, lineno);
        fprintf(stderr,"Error - OCI_CONTINUE\n");
        return (IRRECERR);
    default:
        fprintf(stderr,"Module %s Line %d\n", fname, lineno);
        fprintf(stderr,"Status - %s\n", status);
        return (IRRECERR);
    }
    return (RECOVERERR);
}

```

Appendix - D: RTE Scripts

D.1 RTE Parameters

```
=====+++++ rteparams +++++=====
/* For Oracle in the tpcload program C_LAST =1.          */
/* C-Delta be the difference between C-LOAD and C-Run.    */
/* C-Delta must be a value between 65..119 including the */
/* values of 65 and 119 and excluding the value of 96 and 112 */

#define MASTER_NUM1 1
#define MASTER_NUM2 0

#if MASTER_NUM1
MASTER "master1"
#elif MASTER_NUM2
MASTER "master2"
#endif
/*---- SUT -----*/
SUT="server"
/*-----*/
LASTC=88
MEASUREMENT="1"
WAREHOUSES=16256
/*----- SLAVES -----*/

#if MASTER_NUM1
SLAVES driver1a, driver1b, driver1c, driver1d, driver1e, driver1f,
driver1g, driver1h, driver2a, driver2b, driver2c, driver2d, driver2e,
driver2f, driver2g, driver2h, driver3a, driver3b, driver3c, driver3d,
driver3e, driver3f, driver3g, driver3h, driver4a, driver4b, driver4c,
driver4d, driver4e, driver4f, driver4g, driver4h
#elif MASTER_NUM2
SLAVES driver5a, driver5b, driver5c, driver5d, driver5e, driver5f,
driver5g, driver5h, driver6a, driver6b, driver6c, driver6d, driver6e,
driver6f, driver6g, driver6h, driver7a, driver7b, driver7c, driver7d,
driver7e, driver7f, driver7g, driver7h, driver8a, driver8b, driver8c,
driver8d, driver8e, driver8f, driver8g, driver8h
#endif

/* ---- CLIENTS -----*/
#if MASTER_NUM1
MAIN_CLIENT = client1
CLIENT_REAL = "client1 client2 client3 client4"
#elif MASTER_NUM2
MAIN_CLIENT = client5
CLIENT_REAL = "client5 client6 client7 client8"
#endif

/*---- more client ctuff -----*/
#if MASTER_NUM1
CLIENT client1 tpcc tpcc
CLIENT client2 tpcc tpcc
CLIENT client3 tpcc tpcc
CLIENT client4 tpcc tpcc
#elif MASTER_NUM2
CLIENT client5 tpcc tpcc
CLIENT client6 tpcc tpcc
CLIENT client7 tpcc tpcc
CLIENT client8 tpcc tpcc
#endif
/*-----*/
TELNET telnet 23
SOCKET socket 199703
/* --- Sockets -----*/
#if MASTER_NUM1
SOCKET_NETWORK socket1 80 driver1a
SOCKET_NETWORK socket2 80 driver1b
SOCKET_NETWORK socket3 80 driver1c
SOCKET_NETWORK socket4 80 driver1d
```

```
SOCKET_NETWORK socket5 80 driver1e
SOCKET_NETWORK socket6 80 driver1f
SOCKET_NETWORK socket7 80 driver1g
SOCKET_NETWORK socket8 80 driver1h

SOCKET_NETWORK socket33 80 driver2a
SOCKET_NETWORK socket34 80 driver2b
SOCKET_NETWORK socket35 80 driver2c
SOCKET_NETWORK socket36 80 driver2d
SOCKET_NETWORK socket37 80 driver2e
SOCKET_NETWORK socket38 80 driver2f
SOCKET_NETWORK socket39 80 driver2g
SOCKET_NETWORK socket40 80 driver2h

SOCKET_NETWORK socket65 80 driver3a
SOCKET_NETWORK socket66 80 driver3b
SOCKET_NETWORK socket67 80 driver3c
SOCKET_NETWORK socket68 80 driver3d
SOCKET_NETWORK socket69 80 driver3e
SOCKET_NETWORK socket70 80 driver3f
SOCKET_NETWORK socket71 80 driver3g
SOCKET_NETWORK socket72 80 driver3h

SOCKET_NETWORK socket97 80 driver4a
SOCKET_NETWORK socket98 80 driver4b
SOCKET_NETWORK socket99 80 driver4c
SOCKET_NETWORK socket100 80 driver4d
SOCKET_NETWORK socket101 80 driver4e
SOCKET_NETWORK socket102 80 driver4f
SOCKET_NETWORK socket103 80 driver4g
SOCKET_NETWORK socket104 80 driver4h
#elif MASTER_NUM2
SOCKET_NETWORK socket1 80 driver5a
SOCKET_NETWORK socket2 80 driver5b
SOCKET_NETWORK socket3 80 driver5c
SOCKET_NETWORK socket4 80 driver5d
SOCKET_NETWORK socket5 80 driver5e
SOCKET_NETWORK socket6 80 driver5f
SOCKET_NETWORK socket7 80 driver5g
SOCKET_NETWORK socket8 80 driver5h

SOCKET_NETWORK socket33 80 driver6a
SOCKET_NETWORK socket34 80 driver6b
SOCKET_NETWORK socket35 80 driver6c
SOCKET_NETWORK socket36 80 driver6d
SOCKET_NETWORK socket37 80 driver6e
SOCKET_NETWORK socket38 80 driver6f
SOCKET_NETWORK socket39 80 driver6g
SOCKET_NETWORK socket40 80 driver6h

SOCKET_NETWORK socket65 80 driver7a
SOCKET_NETWORK socket66 80 driver7b
SOCKET_NETWORK socket67 80 driver7c
SOCKET_NETWORK socket68 80 driver7d
SOCKET_NETWORK socket69 80 driver7e
SOCKET_NETWORK socket70 80 driver7f
SOCKET_NETWORK socket71 80 driver7g
SOCKET_NETWORK socket72 80 driver7h

SOCKET_NETWORK socket97 80 driver8a
SOCKET_NETWORK socket98 80 driver8b
SOCKET_NETWORK socket99 80 driver8c
SOCKET_NETWORK socket100 80 driver8d
SOCKET_NETWORK socket101 80 driver8e
SOCKET_NETWORK socket102 80 driver8f
SOCKET_NETWORK socket103 80 driver8g
SOCKET_NETWORK socket104 80 driver8h

#endif
/*-----*/
#if 1
BEGIN_WAIT=5:00
```

```

RAMPUP=1:15:00
RUNTIME=2:00:00
RAMPDOWN_WAIT=10:00
RAMPDOWN=20:00
#BEGIN_WAIT=5:00
#RAMPUP=45:00
#RUNTIME=1:00:00
#RAMPDOWN_WAIT=20:00
#RAMPDOWN=40:00
#else
BEGIN_WAIT=5:00
RAMPUP=20:00
RUNTIME=40:00
RAMPDOWN_WAIT=5:00
RAMPDOWN=17:00
#endif
INTERVAL=1:00 /* Interval to calculate mix from */
LOGIN_MAX_LOAD = 4
LOGIN_BEGIN = 0 /* skip login state if set to 1 */
LOGIN_TIMEOUT = 600 /* Timeout in seconds for user to login */
NOBEGIN = 1
KEYSTROKE_PACKET_SIZE = 0
MAX_CONCURRENT_SPAWN = 4
SPAWN_COUNT = 25
MIN_PORT = 8088
MAX_PORT = 8089
/* User variables. Think, Emulex Delay, %desired, %min, %max */
#if 1 /* Testing */
NEWORDER = "12.02, 0, 0"
PAYMENT = "12.02, 0, 0, 43.02, 43.02, 43.02 "
ORDSTAT = "10.01, 0, 0, 4.02, 4.02, 4.02 "
DELIVERY = "05.02, 0, 0, 4.02, 4.02, 4.02 "
STOCKLEV = "05.02, 0, 0, 4.02, 4.02, 4.02 "
#elseif 0 /* From rtparams.null */
NEWORDER = "12.25, 0.42, 0.38"
PAYMENT = "12.25, 0.19, 0.23, 43.2, 41.1, 45.3 "
ORDSTAT = "10.50, 0.39, 0.21, 4.1, 3.9, 4.3 "
DELIVERY = "05.5, 0.19, 0.15, 4.1, 3.9, 4.3 "
STOCKLEV = "05.5, 0.25, 0.18, 4.1, 3.9, 4.3 "
#elseif 0 /* From Pookeepsie */
NEWORDER = "16.25, 0.42, 0.38"
PAYMENT = "16.25, 0.19, 0.23, 43.15, 43.15, 43.15 "
ORDSTAT = "14.50, 0.39, 0.21, 4.03, 4.03, 4.03 "
DELIVERY = "09.50, 0.19, 0.15, 4.03, 4.03, 4.03 "
STOCKLEV = "09.50, 0.25, 0.18, 4.03, 4.03, 4.03 "
#endif
/*----- Starting users on sockets -----*/
#if MASTER_NUM1
START_RANGE client1 socket1 2540 1-254
START_RANGE client1 socket2 2540 254-508
START_RANGE client1 socket3 2540 508-762
START_RANGE client1 socket4 2540 762-1016
START_RANGE client1 socket5 2540 1016-1270
START_RANGE client1 socket6 2540 1270-1524
START_RANGE client1 socket7 2540 1524-1778
START_RANGE client1 socket8 2540 1778-2032

START_RANGE client2 socket33 2540 2032-2286
START_RANGE client2 socket34 2540 2286-2540
START_RANGE client2 socket35 2540 2540-2794
START_RANGE client2 socket36 2540 2794-3048
START_RANGE client2 socket37 2540 3048-3302
START_RANGE client2 socket38 2540 3302-3556
START_RANGE client2 socket39 2540 3556-3810
START_RANGE client2 socket40 2540 3810-4064

START_RANGE client3 socket65 2540 4064-4318
START_RANGE client3 socket66 2540 4318-4572
START_RANGE client3 socket67 2540 4572-4826
START_RANGE client3 socket68 2540 4826-5080
START_RANGE client3 socket69 2540 5080-5334
START_RANGE client3 socket70 2540 5334-5588

```

```

START_RANGE client3 socket71 2540 5588-5842
START_RANGE client3 socket72 2540 5842-6096

```

```

START_RANGE client4 socket97 2540 6096-6350
START_RANGE client4 socket98 2540 6350-6604
START_RANGE client4 socket99 2540 6604-6858
START_RANGE client4 socket100 2540 6858-7112
START_RANGE client4 socket101 2540 7112-7366
START_RANGE client4 socket102 2540 7366-7620
START_RANGE client4 socket103 2540 7620-7874
START_RANGE client4 socket104 2540 7874-8128

```

```

#elif MASTER_NUM2
START_RANGE client5 socket1 2540 8128-8382
START_RANGE client5 socket2 2540 8382-8636
START_RANGE client5 socket3 2540 8636-8890
START_RANGE client5 socket4 2540 8890-9144
START_RANGE client5 socket5 2540 9144-9398
START_RANGE client5 socket6 2540 9398-9652
START_RANGE client5 socket7 2540 9652-9906
START_RANGE client5 socket8 2540 9906-10160

```

```

START_RANGE client6 socket33 2540 10160-10414
START_RANGE client6 socket34 2540 10414-10668
START_RANGE client6 socket35 2540 10668-10922
START_RANGE client6 socket36 2540 10922-11176
START_RANGE client6 socket37 2540 11176-11430
START_RANGE client6 socket38 2540 11430-11684
START_RANGE client6 socket39 2540 11684-11938
START_RANGE client6 socket40 2540 11938-12192

```

```

START_RANGE client7 socket65 2540 12192-12446
START_RANGE client7 socket66 2540 12446-12700
START_RANGE client7 socket67 2540 12700-12954
START_RANGE client7 socket68 2540 12954-13208
START_RANGE client7 socket69 2540 13208-13462
START_RANGE client7 socket70 2540 13462-13716
START_RANGE client7 socket71 2540 13716-13970
START_RANGE client7 socket72 2540 13970-14224

```

```

START_RANGE client8 socket97 2540 14224-14478
START_RANGE client8 socket98 2540 14478-14732
START_RANGE client8 socket99 2540 14732-14986
START_RANGE client8 socket100 2540 14986-15240
START_RANGE client8 socket101 2540 15240-15494
START_RANGE client8 socket102 2540 15494-15748
START_RANGE client8 socket103 2540 15748-16002
START_RANGE client8 socket104 2540 16002-16256

```

```

#endif
/*-----*/
#define TES_FLAG_TRACE 0x00000010
#define TES_FLAG_KEYSTROKE_TIME 0x000000200
#define TES_FLAG_LOCAL_LOG 0x000000400
#define TES_FLAG_LOCAL_TRACE 0x000000800
#define TES_FLAG_LOCAL_IPRINT 0x000004000
#if 0
/* SETFLAG ALL TES_FLAG_TRACE */
SETFLAG ALL TES_FLAG_LOCAL_TRACE
SETFLAG ALL TES_FLAG_LOCAL_IPRINT
#endif
#if 0
SETFLAG client31 telnet 1 TES_FLAG_KEYSTROKE_TIME
#endif

```

D.2 RTE Scripts

tpccWeb.h

```
/*
*****
** Project   : AIX TPC-C
** Component : TPC-C/Client
** Name      : tpccWeb.h
** Title     : rte web defines
*****
** Copyright (c) IBM US - AUSTIN 2000
** Classification : IBM Internal Use Only
**
** History   :
**           : Develop by Austin RISC/6000 Performance Team
**
** Comments  :
**
*****
*/

// Transaction Codes
// Transaction Request URLs

#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 TXN_NORD "nord"
#define TXN_PYMT "pymt"
#define TXN_ORDS "ords"
#define TXN_DLVY "dlvy"
#define TXN_STOK "stok"
#define TXN_SEXIT "exit"
#define TXN_MENU "menu"

#define ITEM_CMD_ID_START 11
#define ITEM_CMD_ID_END 55

#define APP_NAME "tpcc"

// Transaction Result Search Strings

#define LOGIN_TITLE "Home Page"
#define MENU_TITLE "Main Menu"
#define NORD_TITLE "New Order"
#define PYMT_TITLE "Payment"
#define ORDS_TITLE "Order Status"
#define DLVY_TITLE "Delivery"
#define STOK_TITLE "Stock Level"

#define LOGIN_TITLE_LEN 9
#define MENU_TITLE_LEN 9
#define NORD_TITLE_LEN 9
#define PYMT_TITLE_LEN 7
#define ORDS_TITLE_LEN 12
#define DLVY_TITLE_LEN 8
```

```
#define STOK_TITLE_LEN 10

#define NORD_RESULTS_TITLE "New Order Results"
#define PYMT_RESULTS_TITLE "Payment Results"
#define ORDS_RESULTS_TITLE "Order Status Results"
#define DLVY_RESULTS_TITLE "Delivery Results"
#define STOK_RESULTS_TITLE "Stock Level Results"

#define NORD_RESULTS_TITLE_LEN 17
#define PYMT_RESULTS_TITLE_LEN 15
#define ORDS_RESULTS_TITLE_LEN 20
#define DLVY_RESULTS_TITLE_LEN 16
#define STOK_RESULTS_TITLE_LEN 19

#define CONTENT_LENGTH "Content-Length: "
#define HEADER_TERMINATOR "\r\n\r\n"

// Field Lengths
// Transaction Request URLs

#define HEADER_TERMINATOR_LENGTH 4
#define CONTENT_LENGTH_STR_LEN 16

// Transaction Request URLs

#define GET_REQUEST "GET %s HTTP/1.1\r\nHost: %s\r\nConnection: Keep-Alive\r\nAccept: text/*\r\n\r\n"
#define GET_REQUEST_EXIT "GET %s HTTP/1.1\r\nHost: %s\r\nConnection: close\r\nAccept: text/*\r\n\r\n"

#define LOGIN_URL "/tpcc/tpcc.html"
#define MENU_URL "/tpcc/tpcc.html?00=menu&02=%d&03=%d"

#define NEW_ORDER_FORM_URL "/tpcc/tpcc.html?00=nord&01=%d"
#define PAYMENT_FORM_URL "/tpcc/tpcc.html?00=pymt&01=%d"
#define ORDER_STATUS_FORM_URL "/tpcc/tpcc.html?00=ords&01=%d"
#define DELIVERY_FORM_URL "/tpcc/tpcc.html?00=dlvy&01=%d"
#define STOCK_FORM_URL "/tpcc/tpcc.html?00=stok&01=%d"
#define EXIT_FORM_URL "/tpcc/tpcc.html?00=exit&01=%d"

#define NEW_ORDER_RESULTS_URL "/tpcc/tpcc.html?00=nord&01=%d&03=%d&04=%d&05=%d&07=%d&08=%d.%02.2d"
#define PAYMENT_RESULTS_CID_URL "/tpcc/tpcc.html?00=pymt&01=%d&03=%d&04=%d&05=%d&07=%d&08=%d.%02.2d"
#define PAYMENT_RESULTS_CLAST_URL "/tpcc/tpcc.html?00=pymt&01=%d&03=%d&04=%d&05=%s&06=%d&07=%d&08=%d.%02.2d"
#define ORDER_STATUS_RESULTS_CLAST_URL "/tpcc/tpcc.html?00=ords&01=%d&03=%d&04=%d&05=%s"
#define ORDER_STATUS_RESULTS_CID_URL "/tpcc/tpcc.html?00=ords&01=%d&03=%d&04=%d&05="
#define DELIVERY_RESULTS_URL "/tpcc/tpcc.html?00=dlvy&01=%d&10=%d"
#define STOCK_RESULTS_URL "/tpcc/tpcc.html?00=stok&01=%d&09=%d"

#define NEW_ORDER_ITEM "&%d=%d"
#define NEW_ORDER_ITEM_ENTRY "&%d=%d&%d=%d&%d=%d"
#define NEW_ORDER_EMPTY_ITEM "&%d="
```

user_master.C

```
/******  
/* user_master.C                      Audit: 05/30/96 */  
/******  
  
static char *rcsid="$Id: user_master.C,v 1.1 1999/02/22 06:31:05  
channui Exp $";  
  
#include <iostream.h>  
#include <stdio.h>  
#include <strings.h>  
#include <stdlib.h>  
#include <unistd.h>  
#define _H_CUR01  
#include <cur00.h>  
#undef _H_CUR01  
extern "C" {  
#include "data/cur01.h"  
int wrefresh (WINDOW *);  
int wclrtoeol(WINDOW *);  
int setupterm(char*,FILE*,int*);  
int nodelay(int);  
int keypad(int);  
int wgetch(WINDOW *);  
}  
#include "data/rte.h"  
#include "data/Stats.h"  
#include "data/misc.h"  
#include "user_tpc.h"  
  
struct header_s {  
    int slave;  
    int num;  
    int type;  
    int num_timestamps;  
    int user_data_length;  
    int data_type;  
};  
  
char *get_variable(char *name);  
int get_variable(char *name, int *number);  
int send_global_data(void);  
int make_ratios (double *buffer);  
extern int ramp_up_complete;  
extern int interval_start_time, interval_stop_time;  
//extern "C" int strcasecmp(char *s1, char *s2);  
//extern "C" int strncasecmp(char *s1, char *s2, int n);  
  
struct UserSpawnData {  
    int Warehouse;  
    int District;  
};  
  
/* user_master.C */  
int user_statistics_print(void);  
// int user_spawn(int *length, char *buffer);  
int user_spawn(int min, int max, int number, int *length, char  
*buffer);  
int user_finished(int length, char *buffer);  
  
extern SlaveStatus slave_status[MAX_SLAVES];  
  
extern Stats status[MAX_TRAN_TYPE][MAX_TIMES];  
extern WINDOW *statistics_win;  
extern UserGlobal *shmglobal;  
  
/* Transaction mix parameters */  
double ratio_desired[6], ratio_min[6], ratio_max[6], ratio_range[6];
```

```
char *ratio_names[] = { "RTE", "NEWORDER", "PAYMENT",  
"ORDSTAT", "DELIVERY",  
"STOCKLEV", NULL };  
char *Status_Names[] = {"Menu", "Keying", "Response", "Think"};
```

```
char *transaction_names[] = { "RTE", "New Order", "Payment",  
"Order Stat",  
"Delivery", "Stock Level", NULL };
```

```
static int current_status = 2, status_needs_refresh = 1;
```

```
int user_statistics_print(void) {  
    int i;  
    static int count = 0;  
    double ratios[6];  
    if (status_needs_refresh) {  
        count = 0;  
        status_needs_refresh = 0;  
        wmove (statistics_win, 0, 0);  
        wprintw (statistics_win, "%11s %8s %8s %8s %8s %6s %6s  
%6s",  
            Status_Names[current_status], "90%", "Avg", "Min", "Max",  
            "Samples", "Ratio", "Mix", "Think");  
    }  
    make_ratios(ratios);
```

```
    for (i = 1; i <= 5; i++) {  
        /* The reason we do this is because calculating the percentiles  
        is expensive */  
        if (count % 10 == 0) {  
            wmove (statistics_win, i, 0);  
            wprintw (statistics_win, "%11s %8.2f",  
                transaction_names[i],  
                status[i][current_status].ninety()/1000.0);  
            count = 0;  
        }  
        wmove (statistics_win, i, 21);  
        wprintw (statistics_win, "%8.2f %8.2f %8.2f %8d %6.2f %6.2f  
%6.2f",  
            status[i][current_status].average()/1000.0,  
            status[i][current_status].min()/1000.0,  
            status[i][current_status].max()/1000.0,  
            status[i][current_status].samples(),  
            ratios[i], shmglobal->chances[i],  
            status[i][3].average()/1000.0);  
    }  
    wmove (statistics_win, 7, 0);  
  
    extern int runtime_counts[MAX_TRAN_TYPE];  
    extern int begin_time, ramp_up, run_time;  
    int start = interval_start_time;  
    int stop = interval_stop_time;  
    double interval = ((double)(stop-start) / (1000*60));  
    double samples = status[1][2].samples();  
    if (interval <= 0 || samples <= 0) {  
        wprintw (statistics_win, "TPM-C: %7s / ", "-----");  
    } else {  
        wprintw (statistics_win, "TPM-C: %7.2f / ", samples/interval);  
    }  
    samples = runtime_counts[1];  
    if (samples > 0) {  
        start = begin_time+((ramp_up>=0)?ramp_up:0);  
        if (run_time > 0 && stop > begin_time + ramp_up + run_time) {  
            stop = begin_time + ramp_up + run_time;  
        }  
        interval = (double)(stop - start)/(1000.0*60.0);  
        wprintw (statistics_win, "%7.2f", samples/interval);  
    } else {  
        wprintw (statistics_win, "-----");  
    }  
}
```

```

    count++;
    return RTE_OK;
}

extern int login_begin;
int login_max_load;

#ifdef WHSEARRAYDBG
int outofboundwarn;
#endif

extern int min_warehouse;
extern int max_warehouse;

const int MAX_WAREHOUSES=100000;
/* All of this 10 stuff is district size. Should be a constant.
   Maybe fix that later */
int num_warehouses = -1;
int warehouses[MAX_WAREHOUSES*10];
int user_spawn(int min, int max, int number, int *length, char *buffer)
{
    //int user_spawn(int number, int *length, char *buffer) {
    int i, min_index;
    int adj_wh = num_warehouses; // adjusted warehouse number
    UserSpawnData *ptr = (UserSpawnData *)buffer;
    *length = sizeof(*ptr);

    // min_index = 0;
    // for (i = 1; i < (num_warehouses)*10 && i <
    MAX_WAREHOUSES*10; i++) {
    //
    // if both min and max are zero, running START, otherwise running
    // START_RANGE. Must also determine what the ending
    // warehouse number
    // will be for said range
    //

    if (min == 0 && max == 0) {
        min++;
        min_index = 0;
    } else {
        adj_wh = max; // inclusive range of wh-s
        min = min * 10;
        min_index = min;
    }
    for (i = min; i < (adj_wh)*10 && i <
    ((MAX_WAREHOUSES+min_warehouse)*10); i++) {
        if (warehouses[i - (min_warehouse*10)] < warehouses[min_index -
    (min_warehouse*10)]) {
            min_index = i;
        }
    }

    ptr->Warehouse = min_index / 10 + 1;
    ptr->District = min_index % 10 + 1;
#ifdef WHSEARRAYDBG
    if ((min_index - (min_warehouse*10) < 0) || (min_index -
    (min_warehouse*10) >= (MAX_WAREHOUSES*10))) {
        if (outofboundwarn) {
            iprint (IPRINT_INFO, "(spawn) Out of range warehouse number
            %d, (%d-%d (start) = %d (rel. num)\n",
            min_index, min_index, min_warehouse, min_index -
            (min_warehouse*10));
            outofboundwarn=0;
        }
    }
#endif

    warehouses[min_index - (min_warehouse*10)]++;
    /* iprint (IPRINT_INFO, "Driver for Warehouse %d, District %d
    started. warehouses[%d]++ = %d\n",

```

```

    ptr->Warehouse, ptr->District, min_index,
    warehouses[min_index - (min_warehouse*10)]); */
    return RTE_OK;
}

int user_finished(int length, char *buffer) {
    UserSpawnData *ptr = (UserSpawnData *)buffer;
    int temp = (ptr->Warehouse-1)*10+ptr->District-1;

#ifdef WHSEARRAYDBG
    if ((temp - min_warehouse*10 < 0) || (temp - min_warehouse*10
    >= MAX_WAREHOUSES*10)) {
        if (outofboundwarn) {
            iprint (IPRINT_INFO, "(finish) Out of range warehouse number
            %d, (%d-%d (start) = %d (rel. num)\n",
            min_index, min_index, min_warehouse, min_index -
            (min_warehouse*10));
            outofboundwarn=0;
        }
    }
#endif

    warehouses[temp - (min_warehouse*10)]--;
    /* iprint (IPRINT_INFO, "Driver for Warehouse %d, District %d
    died. warehouses[%d]-- = %d\n",
    ptr->Warehouse, ptr->District, temp, warehouses[temp -
    (min_warehouse*10)]); */
    return RTE_OK;
}

double limit(double min, double max, double val) {
    if (val < min)
        return min;
    if (val > max)
        return max;
    return val;
}

int make_ratios (double *buffer) {
    int neword = status[NEWORDER][0].samples();
    int payment = status[PAYMENT][0].samples();
    int ordstat = status[ORDSTAT][0].samples();
    int delivery = status[DELIVERY][0].samples();
    int stocklev = status[STOCKLEV][0].samples();
    int total = neword + payment + ordstat + delivery + stocklev;
    int i;

    if (total == 0) {
        buffer[NEWORDER] = 100.0;
        for (i = 2; i < 6; i++) {
            buffer[i] = ratio_desired[i];
            buffer[NEWORDER] -= buffer[i];
        }
        return 0;
    }

    buffer[PAYMENT] = (double)payment / (double)total * 100.0;
    buffer[ORDSTAT] = (double)ordstat / (double)total * 100.0;
    buffer[DELIVERY] = (double)delivery / (double)total * 100.0;
    buffer[STOCKLEV] = (double)stocklev / (double)total * 100.0;
    buffer[NEWORDER] = 100.0 - buffer[PAYMENT] -
    buffer[ORDSTAT] -
    buffer[DELIVERY] - buffer[STOCKLEV];

    return total;
}

int user_global_update(int *length, char *buffer) {
    UserGlobal *shmglobal = (UserGlobal *)buffer;
    static double last[6];

```

```

static last_test_state = 0;
static int users_last=-1;
double ratios[6];
double current[6];
int i, different = 0;
int desired = 0;
int host_busy, all_zero;

*length = sizeof(*shmglobal);

make_ratios(ratios);

/* Calculate ratios we want for next time */
/* Note: we just keep on with the desired values until ramp-up is
complete
    this at least starts us out without any humps or spikes in the
    graph */
if (ramp_up_complete) {
current[NEWORDER] = 100.0;
for (i = 2; i < 6; i++) {
    if (ratio_desired[i] > ratios[i]) {
current[i] = ratio_max[i];
    } else {
current[i] = 2*ratio_desired[i] - ratios[i];
    if (current[i] < ratio_min[i])
current[i] = ratio_min[i];
    }
current[NEWORDER] -= current[i];
}
} else {
for (i = 1; i < 6; i++) {
current[i] = ratio_desired[i];
}
}

/* Add up all the users */
/* This needs to be changed to be more transparent */
shmglobal->total_users = 0;
for (i = 0; i < MAX_SLAVES; i++) {
shmglobal->total_users += slave_status[i].active;
desired      += slave_status[i].desired;
}
/* Count up number of warehouses we WANT to have */
if (num_warehouses < 0) {
num_warehouses = (desired-1)/10+1;
}
shmglobal->max_warehouses = num_warehouses;

host_busy = 0;
all_zero = 1;
for (i = 1; i <= 5; i++) {
if (status[i][current_status].average() != 0) {
all_zero = 0;
}
}
if ( status[i][current_status].average()/1000.0 > login_max_load ) {
host_busy = 1;
}
}
if (shmglobal->host_busy && all_zero) {
host_busy = 1;
}

if (host_busy != shmglobal->host_busy) {
shmglobal->host_busy = host_busy;
different = 1;
}

for (i = 2; i < 6; i++) {
if (current[i] != last[i])
different = 1;
}

```

```

if (last_test_state != shmglobal->test_state) {
different = 1;
last_test_state = shmglobal->test_state;
}

// Don't send if it's the same as last time
if ( !different && shmglobal->total_users == users_last ) {
return RTE_ERROR;
}

users_last = shmglobal->total_users;
for (i = 1; i < 6; i++) {
shmglobal->chances[i] = last[i] = current[i];
}

return RTE_OK;
}

int user_isbusy() {
return shmglobal->host_busy;
}

int parse_array(char *string, int max, int *buffer) {
int i, rc;
char *ptr;
char *temp = strdup(string);
ptr = strtok(temp, ",");
for (i = 0; ptr && i < max; i++) {
rc = sscanf(ptr, "%d", &buffer[i]);
if (rc < 1) {
free(temp);
return i;
}
}
ptr = strtok(NULL, ",");
}
free(temp);
return i;
}

int parse_array(char *string, int max, double *buffer) {
int i, rc;
char *ptr;
char *temp = strdup(string);
ptr = strtok(temp, ",");
for (i = 0; ptr && i < max; i++) {
rc = sscanf(ptr, "%lf", &buffer[i]);
if (rc < 1) {
free(temp);
return i;
}
}
ptr = strtok(NULL, ",");
}
free(temp);
return i;
}

int user_init() {
double dbuffer[32];
int rc, i;
char *ptr;

if (get_variable("KEYSTROKE_SLEEP", &shmglobal-
>keystroke_sleep) != RTE_OK) {
shmglobal->keystroke_sleep = 0;
}
if (get_variable("LOGIN_TIMEOUT", &shmglobal-
>login_timeout) != RTE_OK) {
shmglobal->login_timeout = 120; /* 2 minutes */
}
}

```



```

    if (get_variable("KEYSTROKE_PACKET_SIZE", &shmglobal-
>keystroke_packet_size) != RTE_OK) {
        shmglobal->keystroke_packet_size = 0;
    }
    shmglobal->login_timeout *= 1000;
    if (get_variable("LOGIN_MAX_LOAD", &login_max_load) !=
RTE_OK) {
        login_max_load = 2;
    }
    if (get_variable("WAREHOUSES", &num_warehouses) !=
RTE_OK) {
        num_warehouses = -1;
    }
    if (get_variable("LASTC", &shmglobal->lastc) != RTE_OK) {
        shmglobal->lastc = 193; /* 2 minutes */
    }
    iprint(IPRINT_INFO, "Login Timeout = %s\n",
mstoa_withfrac(shmglobal->login_timeout, 0));
    iprint(IPRINT_INFO, "Keystroke Sleep = %s\n",
mstoa_withfrac(shmglobal->keystroke_sleep*1000, 0));
    iprint(IPRINT_INFO, "Keystroke Packet Size= %d\n", shmglobal-
>keystroke_packet_size);
    if (num_warehouses >= 0) {
        iprint(IPRINT_INFO, "Fixed Warehouses to = %d\n",
num_warehouses);
    }

    if (!(ptr = get_variable("NEWORDER"))) {
        iprint_error ("Error. NEWORDER variable not found\n");
        exit (1);
    }
    if (parse_array(ptr, 3, dbuffer)!=3) {
        iprint_error ("Error. NEWORDER should be think, emulex_menu,
emulex_response");
        exit (1);
    }
    shmglobal->think      [NEWORDER] = dbuffer[0];
    shmglobal->emulex_menu [NEWORDER] = dbuffer[1];
    shmglobal->emulex_response[NEWORDER] = dbuffer[2];
    shmglobal->test_state = 0;

    for (i = 2; i < 6; i++) {
        if (!(ptr = get_variable(ratio_names[i])) ||
(parse_array(ptr, 6, dbuffer)!=6)) {
            iprint(_FILE_, _LINE_, IPRINT_ERROR,
"Error. %s should be think, emulex_menu,
emulex_response, desired, min, max",
ratio_names[i]);
            exit (1);
        }
        shmglobal->think[i]      = dbuffer[0];
        shmglobal->emulex_menu[i] = dbuffer[1];
        shmglobal->emulex_response[i] = dbuffer[2];
        ratio_desired[i]        = dbuffer[3];
        ratio_min[i]             = dbuffer[4];
        ratio_max[i]             = dbuffer[5];
        ratio_range[i]           = ratio_max[i]-ratio_min[i];
    }

    for (i=0; i < (MAX_WAREHOUSES*10); i++) {
        warehouses[i] = 0;
    }

#ifdef WHSEARRAYDBG
    outofboundwarn=1;
#endif

    return RTE_OK;
}

int user_extra_data(header_s *header) {
    int i;

```

```

        int num_timestamps;

        if (header->data_type != RTE_ITEM_KEYSTROKE_TIMES)
            return RTE_OK;
        int *times = (int *)((char *)header+sizeof(struct header_s));
        num_timestamps = header->user_data_length / 4 - 1;

        iprint (IPRINT_TRACE, "Keystroke times = ");
        for (i = 0 ; i < num_timestamps; i++) {
            iprint (IPRINT_TRACE, "%d ", times[i]);
        }
        iprint (IPRINT_TRACE, "\n", times[i]);

        return RTE_OK;
    }

int user_process_command(char *command) {
    char buffer[256], *ptr;
    int i, found, len;
    strncpy (buffer, command, 256);
    ptr = strtok (buffer, " \t");
    found = 0;
    printf ("user_process_command('%s')\n", ptr);
    if (!strncasecmp (ptr, "pause")) {
        shmglobal->test_state = 1;
    } else if (!strncasecmp (ptr, "warmup")) {
        shmglobal->test_state = 2;
    } else if (!strncasecmp (ptr, "notest")) {
        shmglobal->test_state = 0;
    } else if (!strncasecmp (ptr, "login_max_load?")) {
        iprint (IPRINT_WARNING, "Current LOGIN_MAX_LOAD = %d\n",
login_max_load);
    } else if (!strncasecmp (command, "login_max_load=", 15)) {
        login_max_load=atoi(command+15);
        iprint (IPRINT_WARNING, "Set LOGIN_MAX_LOAD = %d\n",
login_max_load);
    } else if (!strncasecmp (ptr, "display")) {
        while (ptr && (ptr = strtok(NULL, " \t"))) {
            if (*ptr == '\0')
                continue;
            for (i = 0; i < 5; i++) {
                len = min(strlen(Status_Names[i]), strlen(ptr));
                if (!strncasecmp (ptr, Status_Names[i], len)) {
                    status_needs_refresh = found = 1;
                    current_status = i;
                    return RTE_OK;
                }
            }
            iprint (IPRINT_WARNING, "Unknown type to display: %s\n",
ptr);
        }
    } else {
        iprint (IPRINT_WARNING, "Unknown Command: '%s'\n",
command);
        return RTE_ERROR;
    }
    return RTE_OK;
}

int transaction_process () {
    return RTE_OK;
}

int user_begin() {
    return RTE_OK;
}

/*
void user_make_header(char *buffer) {
    int i;
    struct user_data_header *data = (struct user_data_header
*)buffer;

```

```

}
*/

user_slave.C

/*****
/* user_slave.C                      Audit: 05/30/96 */
*****/

static char *rcsid="$Id: user_slave.C,v 1.1 1999/02/22 06:31:06
channui Exp $";

/*****
***      TPCC FILE FOR ALL USERS      ***
*****/

#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <string.h>
#include <sys/time.h>
#include "rte_slave.h"
#include "user_tpcc.h"

/* This MUST match the corresponding one in client's inout.h file! */
#define TRIGGER "t021"
//define NOSLEEP
// Increased EXPECT_TIMEOUT from 600000 - oz 10/20/97
#define EXPECT_TIMEOUT 6000000
#define KEYWAIT_FUDGE 5000

extern SHM_Slave *shm;
extern TableEntrySlave *shmentry;
extern DriverStatus *status;
extern echo_trace(char *);
extern echo_trace();
extern char *expect_save;

extern char *expect_buffer_return();

const char *SQL_TPERRNO_MESSAGE = "tperrno";
const char *SQL_RTN_MESSAGE = "rtn:";
const char *SQL_FATAL_MESSAGE = "SQL Fatal Error";
const char *ROLLBACK_MESSAGE = "Item number is not
valid";
const char *CUSTOMER_ID_STRING = "Customer: ";

int      WHSEID; /* warehouse number for each users */

/*****
/* The "uniform()" function has range of the absolute value of the
*/
/* difference between the min. and the max values upto
2147483647. */
*****/
/*-----*/
/* NURand      */
/*-----*/
/* A: 255 for C_LAST, 1023 for C_ID, 8191 for OL_I_ID */
/* x: 0 for C_LAST, 1 for C_ID and OL_I_ID */
/* y: 999 for C_LAST, 3000 for C_ID, 100000 for OL_I_ID */
/*-----*/
long
NURand(int A, int x, int y, long cval)
{
    return (((long) uniform((long) 0, (long) A) | (long) uniform((long) x,
(long) y)) + cval) % (y - x + 1)) + x;
}

/*-----*/
/* getname      */
/*-----*/

```

```

/*-----*/
/* generates a random number from 0 to 999 inclusive */
/* a random name is generated by associating a random */
/* string with each digit of the generated number */
/* three strings are concatenated to generate lastname */
/*-----*/
char      *
getname()
{
    char      *last_name_parts[] =
    {
        "BAR",
        "OUGHT",
        "ABLE",
        "PRI",
        "PRES",
        "ESE",
        "ANTI",
        "CALLY",
        "ATION",
        "EING"
    };
    static char  lastname[128];
    int      random_num;

    #if 1
        random_num = NURand(255, 0, 999, shmglobal->lastc);
    #else
        random_num = NURand(255, 0, 999, LASTC);
    #endif
    strcpy(lastname, last_name_parts[random_num / 100]);
    random_num %= 100;
    strcat(lastname, last_name_parts[random_num / 10]);
    random_num %= 10;
    strcat(lastname, last_name_parts[random_num]);
    return (lastname);
}

typedef struct gen_tran_s {
    int invalid;
    void *data;
    long len;
    long keywait;
    long type;
    char *menu;
    char *results_request;
    char *form_request;
} gen_tran_t;

//Joe N.
typedef struct gen_tran_url_s
{
    char *txn_form_url;
    char *txn_results_url;
} gen_tran_url_s;

int generic_transaction( gen_tran_t *data, char *host)
{
    char buffer[2048];
    static int consecutive_errs = 0;
    int rc;
    set_typing_delay(0);
    //iiprint(IPRINT_TRACE, "> generic_transaction sleep (%d)
type(%d) *data (%d)\n", data->type, data->menu, data);
    #ifndef NOSLEEP
        if (shmglobal->test_state == 0)
            transaction_sleep_do();
    #endif

    #ifndef EXPECT_TIMEOUT
        int timeout = EXPECT_TIMEOUT;
    #endif

```

```

#else
    int timeout = 0;
#endif

    // Start the transaction (MENU)
    //iprint(IPRINT_TRACE, "> generic_transaction start (%d)\n", data-
    >type);
    transaction_start(data->type, data->len, data->data);

    //send menu request page
    //iprint(IPRINT_TRACE, "> transmit data->menu: (%s)\n
    request :(%s)",data->menu,data->form_request);
    //iprint(IPRINT_TRACE, "> transmit data->menu: (%s)\n
    request :(%s)",data->menu);
    transmit(data->form_request);

echo_trace ("Waiting for Menu");
switch (*data->menu)
{
    case '1':
        rc = expect_html(NORD_TITLE,timeout,NORD_TITLE_LEN);
        break;
    case '2':
        rc = expect_html(PYMT_TITLE,timeout,PYMT_TITLE_LEN);
        break;
    case '3':
        rc = expect_html(ORDS_TITLE,timeout,ORDS_TITLE_LEN);
        break;
    case '4':
        rc = expect_html(DLVY_TITLE,timeout,DLVY_TITLE_LEN);
        break;
    case '5':
        rc = expect_html(STOK_TITLE,timeout,STOK_TITLE_LEN);
        break;
    default:
        rc = ERROR;
}

if(rc == ERROR)
{
    iprint (IPRINT_ERROR, "Slave %d:Failed to receive %s input
    form\n**Request:-->%s<--\n",
        shmentry->num, data->menu,data->form_request);
    return (ERROR);
}

//if (expect_html(TRIGGER, timeout) == ERROR)
//{
//    iprint (IPRINT_ERROR, "Slave %d: Failed to receive %s
//    screen\n",
//        shmentry->num, data->menu);
//    return (ERROR);
//}

#ifdef NOSLEEP
    usleep(shmglobal->emulex_menu[data->type]*1000000.0+0.9);
#endif

    // Send our request (KEYING)
    transaction_mark(WHERE_NOW);
    echo_trace ("Keying");

#ifdef NOSLEEP
    usleep(data->keywait*1000000+KEYWAIT_FUDGE); // Keying
    delay
#endif
    // Wait for response (RESPONSE)
    transaction_mark(WHERE_NOW);

```

```

    //iprint(IPRINT_TRACE, "> transmit request :(%s)\n",data-
    >results_request);
    transmit(data->results_request);

    echo_trace ("Wait for Response");
    switch (*data->menu)
    {
        case '1':
            rc =
            expect_html(NORD_RESULTS_TITLE,timeout,NORD_RESULTS_T
            ITLE_LEN);
            break;
        case '2':
            rc =
            expect_html(PYMT_RESULTS_TITLE,timeout,PYMT_RESULTS_TI
            TLE_LEN);
            break;
        case '3':
            rc =
            expect_html(ORDS_RESULTS_TITLE,timeout,ORDS_RESULTS_T
            ITLE_LEN);
            break;
        case '4':
            rc =
            expect_html(DLVY_RESULTS_TITLE,timeout,DLVY_RESULTS_TI
            TLE_LEN);
            break;
        case '5':
            rc =
            expect_html(STOK_RESULTS_TITLE,timeout,STOK_RESULTS_TI
            TLE_LEN);
            break;
        default:
            rc = ERROR;
    }

    if(rc == ERROR)
    {
        iprint (IPRINT_ERROR, "Slave %d:Failed to receive %s result
        page\n**Request:-->%s<--\n",
            shmentry->num, data->menu,data->form_request);
        return (ERROR);
    }

    // if (expect_html(TRIGGER, timeout) == ERROR) {
    //iprint (IPRINT_ERROR, "Slave %d: Failed to receive %s
    response\n",
    //    shmentry->num, data->menu);
    //    return (ERROR);
    // }

#ifdef NOSLEEP
    usleep(shmglobal->emulex_response[data-
    >type]*1000000.0+0.9);
#endif

    // Look for errors and set our think time (THINK)
    transaction_mark(WHERE_NOW);
    if (expect_buffer_search("ERROR:",6))
    {
        data->invalid = 1;
        iprint (IPRINT_ERROR, "Slave %d: %s found '%s'\n%s\n",
            shmentry->num, data->menu,
            "ERROR:",expect_buffer_return());
        // Very dangerous, keep going rather than exiting...
        //return RTE_ERROR;
        // Check for consecutive errors and if there are more than
        // 4 of them exit - allow for transient errors to make
        // tuning and testing easier -oz
        // In either case the transaction is marked as invalid and
        // will be reported as an error by the analyze program.
    }

```

```

        if (consecutive_errs++ > 4)
            return RTE_ERROR;
    }
    else
    {
        consecutive_errs = 0;
    }
    echo_trace ("Thinking");
    transaction_sleep_set(neg_exp_4(shmglobal->think[data-
>type])*1000.0);
    //iprint(IPRINT_TRACE, "< generic_transaction finish\n");
    return (RTE_OK);
}

/*****
***          Delivery Transaction          ***
*****/
int
Delivery(char *host,int terminal)
{
    static struct delivery_struct delivery, delivery_new;
    int      rc;
    char      *ptr;
// char      buffer[256];
    gen_tran_t   tran;

    tran.invalid = 0;
    tran.data    = &delivery;
    tran.len     = sizeof(delivery);
    tran.keywait = 2;
    tran.type    = DELIVERY;
    tran.menu    = "4";
//tran.request = buffer;

    //Joe N.
    char dlvy_url[128];
    char form_buffer[256];
    char results_buffer[256];

    tran.form_request = form_buffer;
    tran.results_request = results_buffer;

    //create dlvy form request
    sprintf(dlvy_url,DELIVERY_FORM_URL,terminal);
    sprintf(form_buffer,GET_REQUEST,dlvy_url,host);

    // Set up all data for new transactions
    delivery_new.carrier = uniform(1, 10); // carrier # 1 to 10

    //create dlvy results request
    sprintf(dlvy_url,DELIVERY_RESULTS_URL,terminal,delivery_new
.carrier);
    sprintf(results_buffer,GET_REQUEST,dlvy_url,host);

    // Now create the actual request
    //ptr = buffer;
    //ptr += sprintf(ptr, "%d\n", delivery_new.carrier);

    // Go do the transaction
    rc = generic_transaction(&tran,host);
    delivery    = delivery_new;
    delivery.invalid = tran.invalid;

    //iprint(IPRINT_TRACE,"dlvy txn finished, rc:%d
tran.invalid:%d\n",rc,delivery.invalid);

    return (rc);
}

/*****
***          New Order Transaction          ***
*****/

```

```

int NewOrder(char *host,int terminal)
{
    static struct neword_struct neword, neword_new;
    int      i, rc, whses, low_whse=1;
//char      buffer[2048];

    //Joe N.
    char nord_form_url[128];
    char form_buffer[512];

    char nord_results_url[2048];
    char results_buffer[4096];

    char      *ptr;
    char      *ptr2;
    const char *err_ptr;

    gen_tran_t   tran;

    tran.invalid    = 0;
    tran.data       = &neword;
    tran.len        = sizeof(neword);
    tran.keywait    = 18;
    tran.type       = NEWORDER;
    tran.menu       = "1";

    //Joe N.
    tran.form_request = form_buffer;
    tran.results_request = results_buffer;

    neword_new.rollback=0;

    /*** SECTION TO DETERMINE ROLLBACK TRANSACTION FOR
    1% OF NEW ORDERS ***/
    neword_new.did    = uniform(1, 10);           // district number
    neword_new.cid    = NURand(1023, 1, 3000, CUSTC); //
customer # 1 to 3000
    neword_new.nloop  = uniform(5, 15);           // number of
items to order (5-15)
    neword_new.olremote = 0;                       // find total number
of remote order-lines

    whses = shmglobal->max_warehouses;

    for (i = 0; i < neword_new.nloop; i++)
    {
        // Warehouse Number
        neword_new.item[i].olswid = WHSEID;
        if (whses > 1 && (uniform(0.0, 100.0) < 1.0))
        {
            /* for 1% of items (if * uniform()==0) */
            /* Generate a uniform whse number that's different from
            WHSEID */
            neword_new.item[i].olswid =
(long) uniform((long) low_whse, (long)whses-1);
            if (neword_new.item[i].olswid >= WHSEID)
                neword_new.item[i].olswid++;
            neword_new.olremote++; // find total number of remote order-
lines
        }
        // Item number 1-100000
        neword_new.item[i].oliid = NURand(8191, 1, 100000, ITEMC);
        // Quantity 1-10
        neword_new.item[i].olquantity = uniform(1, 10);
    }
    /* end of for n_loop */

    // We occasionally force a transaction to have invalid data to force
a
    // rollback
    if (uniform(1, 5000) <= 50)
        neword_new.item[neword_new.nloop-1].oliid = 999999;

```

```

neword_new.oremove = (neword_new.olremove > 0);

//create new order form request
sprintf(nord_form_url,NEW_ORDER_FORM_URL,terminal);

//create get form request
sprintf(form_buffer,GET_REQUEST,nord_form_url,host);

//create new order results url
char itemString[1024];
ptr2=itemString;
short item_cmd_start = ITEM_CMD_ID_START;
for (i = 0; i < neword_new.nloop; i++)
{
    //ptr2 += sprintf(ptr, NEW_ORDER_ITEM_ENTRY,
    // item_cmd_start++,
    // neword_new.item[i].olswid,
    // item_cmd_start++,
    // neword_new.item[i].oliid,
    // item_cmd_start++,
    // neword_new.item[i].olquantity);

    ptr2 += sprintf(ptr2, NEW_ORDER_ITEM,
        item_cmd_start++,
        neword_new.item[i].olswid);

    ptr2 += sprintf(ptr2,NEW_ORDER_ITEM,
        item_cmd_start++,
        neword_new.item[i].oliid);

    ptr2 += sprintf(ptr2, NEW_ORDER_ITEM,
        item_cmd_start++,
        neword_new.item[i].olquantity);
}
//seal up url w/ empty items
for (i = item_cmd_start;i <= ITEM_CMD_ID_END; i++)
{
    ptr2 += sprintf(ptr2,NEW_ORDER_EMPTY_ITEM,i);
}

sprintf(nord_results_url,NEW_ORDER_RESULTS_URL,terminal,
        neword_new.did,neword_new.cid,
        itemString);

//create get results request
sprintf(results_buffer,GET_REQUEST,nord_results_url,host);

//ptr = buffer;
//ptr += sprintf(ptr, "%d\t%d", neword_new.did, neword_new.cid);
// for (i = 0; i < neword_new.nloop; i++)
// {
//     ptr += sprintf(ptr, "\t%d\t%d\t%d",
//     // neword_new.item[i].olswid,
//     // neword_new.item[i].oliid,
//     // neword_new.item[i].olquantity);
// }
// ptr += sprintf(ptr, "\n");

// Go do the transaction
rc = generic_transaction(&tran,host);
neword = neword_new;
neword.invalid = tran.invalid;

// Check for a rollback
if ((err_ptr = expect_buffer_search("Item number is not valid",24)))
{
    neword.rollback=1;
    echo_trace ("Found rollback!\n");
}

```

```

// Grab the orderID from the
if (!(err_ptr = expect_buffer_search("Order Number: ",14)))
{
    echo_trace ("Didn't find order-id for neworder");
    iprint (IPRINT_ERROR, "Neworder didn't have Order-
ID\n%s\n",expect_buffer_return());
    //iprint (IPRINT_ERROR, "Neworder didn't have Order-ID\n");
    neword.oid = -1;
}
else
{
    neword.oid = atoi(err_ptr+14);
    // iprint(IPRINT_ERROR,"New order order
id:%d\n",neword.oid);
}

```

```

// Grab the orderID from the
//if (!(ptr2 = expect_after_match("\033[6;15H")))
//{
//    echo_trace ("Didn't find order-id for neworder");
//    iprint (IPRINT_ERROR, "Neworder didn't have Order-ID");
//    neword.oid = -1;
//} else {
//    neword.oid = atoi(ptr2+8);
//}

// This is really not useful since we aren't going to be sending
individual
// keystrokes anymore
if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME) {
    log_data(RTE_ITEM_KEYSTROKE_TIMES,
        keystroke_length*sizeof(int),keystroke_times);
}

//iprint(IPRINT_TRACE,"nord txn finished, rc:%d
tran.invalid:%d\n",rc,tran.invalid);
return (rc);
}

```

```

/*****
***      Order Status Transaction      ***
*****/
int OrderStatus(char *host,int terminal) {
    static struct ordstat_struct ordstat, ordstat_new;
    //char    buffer[2048];
    int      rc;
    char      *ptr;
    gen_tran_t  tran;

    tran.invalid = 0;
    tran.data    = &ordstat;
    tran.len     = sizeof(ordstat);
    tran.keywait = 2;
    tran.type    = ORDSTAT;
    tran.menu    = "3";
    //tran.request = buffer;

    //Joe N.
    char ords_url[256];
    char form_buffer[512];
    char results_buffer[2048];

    tran.results_request = results_buffer;
    tran.form_request = form_buffer;

    //create order status form request
    sprintf(ords_url,ORDER_STATUS_FORM_URL,terminal);
    sprintf(form_buffer,GET_REQUEST,ords_url,host);
}

```

```

// Set up all data for new transactions
ordstat_new.did = uniform(1, 10); /* district number 1 to 10 */
if (uniform(1, 100) <= 60)
{
    /* for 60% of transactions */
    char *tmp = getname();
    strcpy(ordstat_new.clast, tmp); /* by customer last name */
    if (ordstat_new.clast[0] < 'A' || ordstat_new.clast[0] > 'Z')
    {
        iprint (IPRINT_ERROR,
            "ASSERTION: OrderStatus getname() returns invalid name!
            '%s'\n",
            ordstat_new.clast);
        return RTE_ERROR;
    }
    ordstat_new.byname = 1;
    ordstat_new.cid = 0;
}
else
{
    ordstat_new.cid = NURand(1023, 1, 3000, CUSTC); /* cust. # 1
to 3000 */
    ordstat_new.byname = 0;
    ordstat_new.clast[0] = (char) NULL;
}

//iprint(IPRINT_TRACE,"Order status fields,w_id:%d d_id:%d
n_d_id:%d c_id:%d\n", data->Warehouse,data-
>District,ordstat_new.did,ordstat_new.cid);

//create order status url request
if (ordstat_new.byname)

    sprintf(ords_url,ORDER_STATUS_RESULTS_CLAST_URL,termi
nal,ordstat_new.did,

            ordstat_new.clast);
else

    sprintf(ords_url,ORDER_STATUS_RESULTS_CID_URL,terminal,
ordstat_new.did,

            ordstat_new.cid);

    sprintf(results_buffer,GET_REQUEST,ords_url,host);

//Now create the actual request
//ptr = results_buffer;
//ptr += sprintf(ptr, "%d\t", ordstat_new.did);
//if (ordstat_new.byname)
//{
//    ptr += sprintf(ptr, "\t%s\n", ordstat_new.clast);
//}
//else
//{
//    ptr += sprintf(ptr, "%d\n", ordstat_new.cid);
//}

// Go do the transaction
rc = generic_transaction(&tran,host);
ordstat = ordstat_new;
ordstat.invalid = tran.invalid;

//iprint(IPRINT_TRACE,"ords txn finished, rc:%d
tran.invalid:%d\n",rc,tran.invalid);
return (rc);
}

/*****
***      Payment Transaction      ***
*****/
int
Payment(char *host,int terminal)

```

```

{
    static struct payment_struct payment, payment_new;
    int        dollars, cents, rc, whses, low_whse = 1;
    // char      buffer[2048];
    char        *ptr;
    gen_tran_t   tran;

    tran.invalid = 0;
    tran.data    = &payment;
    tran.len     = sizeof(payment);
    tran.keywait = 3;
    tran.type    = PAYMENT;
    tran.menu    = "2";
    //tran.request = buffer;

    //Joe N.
    char pymt_url[128];
    char form_buffer[256];
    char results_buffer[2048];

    tran.results_request = results_buffer;
    tran.form_request = form_buffer;

    //create pymt form url
    sprintf(pymt_url,PAYMENT_FORM_URL,terminal);
    sprintf(form_buffer,GET_REQUEST,pymt_url,host);

    payment_new.did = uniform(1, 10); /* district number 1 to 10 */
    if (uniform(1, 100) <= 60) { /* for 60% of transactions */
        strncpy(payment_new.clast, getname(), 17); // by customer last
name
        if (payment_new.clast[0] < 'A' || payment_new.clast[0] > 'Z') {
            iprint (IPRINT_ERROR,
                "ASSERTION: payment_new getname() returns invalid
name! '%s'\n",
                payment_new.clast);
            return RTE_ERROR;
        }
        payment_new.byname = 1;
        payment_new.cid = 0;
    } else {
        payment_new.cid = NURand(1023, 1, 3000, CUSTC); /* cust. # 1
to 3000 */
        payment_new.byname = 0;
        payment_new.clast[0] = (char) NULL;
    }

    whses = shmglobal->max_warehouses;

    if (whses < 2 || uniform(1, 100) <= 85) { /* for 85 % of transactions
*/
        payment_new.cwid = WHSEID;
        payment_new.cdid = payment_new.did;
        payment_new.remote = 0;
    } else { /* for 15 % of transactions */
        payment_new.cwid = (long) uniform((long)low_whse, (long)
whses-1);
        if (payment_new.cwid >= WHSEID)
            payment_new.cwid++;

        payment_new.remote = 1;
        payment_new.cdid = uniform(1, 10); /* district 1 to 10 */
    }

    dollars = uniform(1, 5000); /* dollar amt = 1 to 5000 */
    if (dollars == 5000)
        cents = 0;
    else
        cents = uniform(0, 99);
}

```

```

    payment_new.amount = ((double) dollars) + ((double) cents) /
100.0;

    //create payment results url
    if (payment_new.byname)
        sprintf(pymt_url,PAYMENT_RESULTS_CLAST_URL,terminal,

payment_new.did,payment_new.clast,payment_new.cwid,
        payment_new.cdiddollars,cents);
    else
        sprintf(pymt_url,PAYMENT_RESULTS_CID_URL,terminal,

payment_new.did,payment_new.cid,payment_new.cwid,
        payment_new.cdiddollars,cents);
    sprintf(results_buffer,GET_REQUEST,pymt_url,host);

//Now create the actual request
//ptr = buffer;
//ptr += sprintf(ptr, "%d\t", payment_new.did);
//if (payment_new.byname) {
//ptr += sprintf(ptr, "\t%s\t", payment_new.clast);
//} else {
//ptr += sprintf(ptr, "%d\t\t", payment_new.cid);
//}
//ptr += sprintf(ptr, "%d\t%d\t", payment_new.cwid,
payment_new.cdidd);
//ptr += sprintf(ptr, "%d.%02.2d\n", dollars, cents);

    // Go do the transaction

    rc = generic_transaction(&tran,host);
    payment    = payment_new;
    payment.invalid = tran.invalid;

    //iprint(IPRINT_TRACE,"pymt txn finished, rc:%d
tran.invalid:%d\n",rc,tran.invalid);

    return (rc);
}

/*****
***      Stock Level Transaction      ***
*****/
int
StockLevel(char *host,int terminal)
{
    static struct stocklev_struct stocklevel, stocklevel_new;
    //char    buffer[2048];
    int      rc;
    char     *ptr;
    gen_tran_t  tran;

    tran.invalid = 0;
    tran.data    = &stocklevel;
    tran.len     = sizeof(stocklevel);
    tran.keywait = 2;
    tran.type    = STOCKLEV;
    tran.menu    = "5";
    //tran.request = buffer;

    //Joe N.
    char stok_url[128];
    char form_buffer[256];
    char results_buffer[2048];

    tran.results_request = results_buffer;
    tran.form_request = form_buffer;

    //create stok form url
    sprintf(stok_url,STOCK_FORM_URL,terminal);

```

```

    sprintf(form_buffer,GET_REQUEST,stok_url,host);

    stocklevel_new.invalid = 0;
    stocklevel_new.threshold = uniform(10, 20); /* uniform no.
between 10 and
        * 20 */

    //create stok results url
    sprintf(stok_url,STOCK_RESULTS_URL,terminal,stocklevel_new.t
hreshold);
    sprintf(results_buffer,GET_REQUEST,stok_url,host);

// Now create the actual request
//ptr = buffer;
//ptr += sprintf(ptr, "%d\n", stocklevel_new.threshold);

    // Go do the transaction
    rc = generic_transaction(&tran,host);
    stocklevel    = stocklevel_new;
    stocklevel.invalid = tran.invalid;

    //iprint(IPRINT_TRACE,"stok txn finished, rc:%d
tran.invalid:%d\n",rc,tran.invalid);
    return (rc);
}

/*****
*** MAIN() ***
*****/
int
user_transaction(char *host,void *data,int terminal)
{
    UserLocal *localdata = (UserLocal *)data;

    char    logout[32];
    double  ntask;
    int     resp;
    static int task = 0;

    if (shmentry->flags & TES_FLAG_KEYSTROKE_TIME)
    {
        int rc;
        /* Wait for specified period of time */
        sleep (shmglobal->keystroke_sleep);
        /* Quit after one transaction */
        shm->lock(shmentry->pid);
        shmentry->flags |= TES_FLAG_DIE;
        shm->unlock(shmentry->pid);
        rc = NewOrder(host,terminal);
        iprint (IPRINT_INFO, "Slave %d: Keystroke timing setting die
flag\n", shmentry->num);
        return rc;
    }

    #if 1
    switch (shmglobal->test_state)
    {
        case 0: // Normal
            break;
        case 1: // pause
            sleep (1);
            return RTE_OK;
        case 2: // warmup
            switch(task++)
            {
                case 0: return Delivery(host,terminal);
                case 1: return OrderStatus(host,terminal);
                case 2: return Payment(host,terminal);
                case 3: return StockLevel(host,terminal);
                case 4: task = 0; return NewOrder(host,terminal);
                return NewOrder(host,terminal);
            }
    }

```

```

    }
}
/*****
*** CHOOSE ONE OF THE TRANSACTIONS
***
*****/
/*****
ntask = (double) uniform(0.0, 100.0);
if (ntask <= shmglobal->chances[DELIVERY])
{
    return Delivery(host,terminal);
    //return NewOrder(host,terminal);
}
ntask -= shmglobal->chances[DELIVERY];
if (ntask <= shmglobal->chances[ORDSTAT])
{
    return OrderStatus(host,terminal);
}
ntask -= shmglobal->chances[ORDSTAT];
if (ntask <= shmglobal->chances[PAYMENT])
{
    return Payment(host,terminal);
}
ntask -= shmglobal->chances[PAYMENT];
if (ntask <= shmglobal->chances[STOCKLEV])
{
    return StockLevel(host,terminal);
}
return NewOrder(host,terminal);
}

#else
// this code should be shared between all of the users on a slave
// int the best case it should be shared between all of the slaves,
// but that would be too costly.
// for now it is done on a per user basis. If this thing is ever
// modified to be threaded then it will probably go to the per-
process
// basis. Although with shared memory, it would be possible to go
to
// per-slave. Actually, before this code is put into use it must be
// fixed up to share across processes. Right now it will take, on
average,
// 22 minutes for one user to just key in the 100 entries.

// use a card deck with no replacement to fulfill the requirements
{
    int deck[100], count=-1, i, size=1, tmp;
    // lock deck
    if (count < 0) {
        // deck is empty fill it up
        count = 0;
        for (i = 0; i < 43 * size; i++) {
            deck[count++] = Payment;
        }
        for (i = 0; i < 4 * size; i++) {
            deck[count++] = StockLevel;
        }
        for (i = 0; i < 4 * size; i++) {
            deck[count++] = OrderStatus;
        }
        for (i = 0; i < 4 * size; i++) {
            deck[count++] = Delivery;
        }
        for (; count < 100 * size; i++) {
            deck[count++] = NewOrder;
        }
        // randomize the deck
        for (i = 0; i < 100 * size; i++) {
            int tmp;
            int pick = uniform(i+1, 100);
            tmp = deck[i];
            deck[i] = deck[pick];
            deck[pick] = tmp;
        }
    }
}

```

```

    }
}
tmp = deck[count--];
// unlock deck

switch(tmp) {
case Delivery: return Delivery(host,terminal);
case OrderStatus: return OrderStatus(host,terminal);
case Payment: return Payment(host,terminal);
case StockLevel: return StockLevel(host,terminal);
case NewOrder: return NewOrder(host,terminal);
}

/*
switch(tmp) {
case Delivery: return Payment(host,terminal);
case OrderStatus: return Payment(host,terminal);
case Payment: return Payment(host,terminal);
case StockLevel: return Payment(host,terminal);
case NewOrder: return NewOrder(host,terminal);
}
*/
}
#endif

#if 0
if (resp != RTE_OK) { /* logoff if response is not correct */
    strcpy(logout, "9\n"); /* menu option 9 */
    transmit(logout);
    resp = expect("tpcc_cstux_inf:");
    return (ERROR);
} else
    return (RTE_OK);
#endif
} /* end of Main */

int user_parameter_change(void) {
    #if 0
    int i;
    iprint(IPRINT_TRACE, "Slave %d: total_users = %d\n", shmentry-
>num);
    iprint(IPRINT_TRACE, "Slave %d: chances = ", shmentry-
>num);
    for (i = 0; i < MAX_TRAN_TYPE; i++)
        iprint(IPRINT_TRACE, "%6.2f ", shmglobal->chances[i]);
    iprint(IPRINT_TRACE, "\nSlave %d: think = ", shmentry-
>num);
    for (i = 0; i < MAX_TRAN_TYPE; i++)
        iprint(IPRINT_TRACE, "%6.2f ", shmglobal->think[i]);
    iprint(IPRINT_TRACE, "\n");
    #endif
    return RTE_OK;
}

int user_login(char *user, char *password, void *data) {
    UserLocal *localdata = (UserLocal *)data;
    int rc;
    int timeout_value = shmglobal->login_timeout;
    char buffer[32];
    set_typing_delay(0);

    rc = expect (TRIGGER, timeout_value);
    if (rc == RTE_ERROR) {
        iprint(IPRINT_ERROR, "Slave %d: didn't find Warehouse
prompt\n", shmentry->num);
    }

    sprintf(buffer, "%d\t%d\n", localdata->Warehouse, localdata-
>District);
    transmit(buffer);
}

```



```

    iprint (IPRINT_TRACE, "Slave %d: Warehouse=%d, District=%d,
pid=%d\n", shmentry->num, localdata->Warehouse, localdata-
>District, getpid());

    rc = expect (TRIGGER, timeout_value);
    if (rc != RTE_OK) {
        iprint (IPRINT_ERROR, "Slave %d: Failed logging in\n",
shmentry->num);
        return RTE_ERROR;
    }
    return RTE_OK;
}

int user_login_html(char *host,void *data,int *terminal)
{
    UserLocal *localdata = (UserLocal *)data;
    int rc;
    int timeout_value = shmglobal->login_timeout;
    char request[256];
    char url[30];
    set_typing_delay(0);

    //iprint(IPRINT_ERROR,"Generating login request for
host:%s\n",host);

    //generate login page request
    sprintf(request,GET_REQUEST,LOGIN_URL,host);

    //iprint(IPRINT_ERROR,"sending login form request:%s\n",request);

    //send the request
    transmit(request);
    //iprint(IPRINT_ERROR,"login request sent, reading response in
expect_html()\n");

    //read the request
    rc =
    expect_html(LOGIN_TITLE,timeout_value,LOGIN_TITLE_LEN);
    if (rc != RTE_OK)
    {
        iprint(IPRINT_ERROR,"Login request failed, unable to find login
key words:%s\n",LOGIN_TITLE);
        return RTE_ERROR;
    }

    //iprint(IPRINT_ERROR,"login request read\n");

    //generate url and page get request
    sprintf(url,MENU_URL,localdata->Warehouse, localdata->District);
    sprintf(request,GET_REQUEST,url,host);

    //iprint(IPRINT_ERROR,"sending login results
request:%s\n",request);

    transmit(request);

    //iprint (IPRINT_TRACE, "Slave %d: Warehouse=%d, District=%d,
pid=%d\n", shmentry->num, localdata->Warehouse, localdata-
>District, getpid());
    rc = expect_html(MENU_TITLE,
timeout_value,MENU_TITLE_LEN);
    if (rc != RTE_OK)
    {
        iprint (IPRINT_ERROR, "Slave %d: Failed logging in\n",
shmentry->num);
        return RTE_ERROR;
    }
    //iprint (IPRINT_TRACE, "User login successful Slave%d:
Warehouse=%d, District=%d, pid=%d\n",shmentry->num, localdata-
>Warehouse, localdata->District, getpid());

```

```

    rc = get_term_id(terminal);
    if(rc != RTE_OK)
        return RTE_ERROR;

    iprint(IPRINT_TRACE, "Terimnal set for this user:%d w/
warehouse:%d district:%d\n",*terminal,localdata-
>Warehouse,localdata->District);

    return RTE_OK;
}

int get_term_id(int *terminal)
{
    //search for terminal id
    const char *termID_ptr;
    if (!(termID_ptr = expect_buffer_search("NAME=\01"
VALUE="",17)))
    {
        echo_trace ("Did not find terminal id in response...");
        iprint (IPRINT_ERROR, "No terminal id specified.");
        return RTE_ERROR;
    }
    else
    {
        *terminal = atoi(termID_ptr+17);
        iprint(IPRINT_ERROR,"Terminal id:%d\n",terminal);
    }
    return RTE_OK;
}

int user_init () {
    extern int expect_save_active;
    WHSEID = shmlocal->Warehouse;

    status->max_transmit = shmglobal->keystroke_packet_size;
    expect_save_active = 1;
    return RTE_OK;
}

int user_logout () {
    //transmit("9");
    //transmit("GET /tpcc/tpcc.html?00=exit\n\nConnection:
Close\n\n");
    iprint (IPRINT_TRACE, "Slave %d: Warehouse=%d, District=%d
logging out\n", shmentry->num, shmlocal->Warehouse, shmlocal-
>District);
    return RTE_OK;
}

int user_cleanup () {
    transaction_sleep_do();
    transaction_start(0, 0, NULL); // Just something to clear out the
buffer...
    return RTE_OK;
}

int user_spawn_ok() {
    int rc, hb;
    hb = ((UserGlobal *) (shm->global_data))->host_busy;
    rc = hb?RTE_ERROR:RTE_OK;
    return rc;
}

```

user tpcc.h

```

/*****
/* user_tpcc.h                      Audit: 05/30/96 */
*****/

/* $Id: user_tpcc.h,v 1.1 1999/02/22 06:31:06 channui Exp $ */

```

```

#ifndef USER_TPCC_H
#define USER_TPCC_H

#include "data/rte_common.h"

/*****
*** run-time constant for customer last name from 0 to 255,
***
*** run-time constant for customer id from 0 to 1023,
*** run-time constant for item id from 0 to 8191.
*****/
/* #define LASTC 117 */
/* Change for 3.1 */
#define LASTC 193
#define CUSTC 319
#define ITEM_C 3849

/*****
*** response type
*****/
/* #define OK 1 */
/* #define ERROR -1 */

/*****
*** transaction type
*****/
#define NEWORDER 1
#define PAYMENT 2
#define ORDSTAT 3
#define DELIVERY 4
#define STOCKLEV 5

/*****
*** transaction structures
*****/
struct neword_struct {
    char    invalid; /* transaction completed successfully */
    long    did;
    long    cid;
    long    oid; /* Order-ID returned from client */
    long    nloop; /* number of order line, avg = 15 */
    char    oremote; /* 1 for remote order, 10% */
    long    olremote; /* number of remote order line, 1% */
    char    rollback; /* actually saw rollback text on screen */
    struct items_struct {
        long    olswid;
        long    oliid;
        long    olquantity;
    } item[15];
};

struct payment_struct {
    char    invalid; /* transaction completed successfully */
    long    did;
    long    cid;
    long    cwid;
    long    cdid;
    char    clast[17];
    double    amount;
    char    byname; /* 1 for by last name, 0 for by id */
    char    remote; /* 1 for remote warehouse, 0 otherwise */
};

struct ordstat_struct {
    char    invalid; /* transaction completed successfully */
    long    did;
    long    cid;
    char    clast[17];
    char    byname; /* 1 for by last name, 0 for by id */
};

struct delivery_struct {

```

```

    char    invalid; /* transaction completed successfully */
    char    carrier;
};

struct stocklev_struct {
    char    invalid; /* transaction completed successfully */
    long    threshold;
};

struct generic_struct {
    char    invalid; /* transaction completed successfully */
};

typedef union transaction_info {
    char    invalid;
    struct generic_struct    generic;
    struct neword_struct    neword;
    struct payment_struct    payment;
    struct ordstat_struct    ordstat;
    struct delivery_struct    delivery;
    struct stocklev_struct    stocklev;
} transaction_info;

struct UserGlobal {
    int    total_users;
    int    max_warehouses;
    int    keystroke_sleep;
    int    login_timeout;
    int    keystroke_packet_size;
    int    lastc;
    int    test_state;
    int    host_busy;
    double    chances[MAX_TRAN_TYPE];
    double    think[MAX_TRAN_TYPE];
    double    emulex_response[MAX_TRAN_TYPE];
    double    emulex_menu    [MAX_TRAN_TYPE];
};

struct UserLocal {
    int    Warehouse;
    int    District;
};

/*
struct user_data_header {
};
*/

extern struct UserGlobal *shmglobal;
extern struct UserLocal *shmlocal;

#endif

```

Appendix - E: Pricing

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
<http://www.microsoft.com/>

Microsoft

August 2, 2005

IBM Corporation
Tony Petrossian
11501 BURNET ROAD
Austin, TX 78758

Mr. Petrossian:

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
C11-00821	Windows 2000 Server <i>Server License Only - No CALs</i> <i>Discount Schedule: No Level</i> <i>Unit Price reflects a 8% discount from the retail unit price of \$799.</i>	\$738	8	\$5,904
254-00170	Visual C++ Standard Edition <i>Discount Schedule: No Discounts Applied</i>	\$109	1	\$109
	Microsoft Problem Resolution Services <i>Professional Support</i> <i>(1 incident)</i>	\$245	1	\$245

All products are currently orderable through Microsoft's normal distribution channels.

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: PCtope0502088897.

Please include this Reference ID in any correspondence regarding this price quote.



October 13, 2005

Tony Petrossian
IBM Corporation

Dear Tony,
 The requested quote for the p5-570 TPC-C benchmark is below.

Product	Description	Qty	List Price	Extended Purchase	Monthly Maint.	Extended Warr/Mnt
Server Hardware						
9117-570	pSeries p5 9117 Model 570	1	\$2,922.00	\$2,922.00	\$128.00	\$3,072.00
6942-25B	9117-570 24x7x4 WSU	1	\$4,070.00	\$0.00		\$4,074.00
1846	Op Panel	1	\$150.00	\$150.00		
2640	IDE Slimline DVD-ROM Drive	1	\$378.00	\$378.00		
3274	73.4 GB 10,000 RPM Ultra320 SCSI Disk Drive Assembly	2	\$439.00	\$878.00		
	32768MB (4x8192MB) DIMMs, 208-pin, 200MHz Stacked					
4492	DDR1 SDRAM	4	\$103,000.00	\$412,000.00		
4651	Rack Indicator, Rack #1	1	N/C	N/C		
5716	2 Gigabit Fibre Channel PCI-X Adapter	6	\$2,267.00	\$13,602.00		
6458	Power Cable -- Drawer to IBM PDU, 14-foot, 250V/10A	2	\$14.00	\$28.00		
7164	IBM Rack-mount Drawer Rail Kit	1	\$168.00	\$168.00		
7832	Processor, 0/2 Way 1.9 GHz with DDR1 memory	2	\$10,900.00	\$21,800.00	\$320.00	\$7,680.00
7865	CEC Backplane	1	\$1,200.00	\$1,200.00		
7866	I/O Backplane	1	\$4,100.00	\$4,100.00		
7867	Midplane	1	\$500.00	\$500.00		
7868	DASD Backplane	1	\$1,200.00	\$1,200.00		
7869	Media Backplane	1	\$140.00	\$140.00		
7870	Power Midplane	1	\$200.00	\$200.00		
7875	CPU Regulator	3	\$1,100.00	\$3,300.00		
7878	Serial Port Riser Card	1	\$100.00	\$100.00		
7879	System Drawer Enclosure	1	\$350.00	\$350.00		
7881	Service Processor Card (FSP)	1	\$650.00	\$650.00		
7888	AC Power Supply, 240V, 1400 W	2	\$800.00	\$1,600.00		
7898	Single Processor Activation For #7832	2	\$21,800.00	\$43,600.00	\$282.00	\$6,768.00
8454	Zero-priced Processor Activation Code for #7832	2	N/C	N/C	\$282.00	\$6,768.00
9300	Language Group Specify - US English	1	N/C	N/C		
			Subtotal	\$508,866.00	Subtotal	\$28,362.00
7014-T00	Rack 1:Rack Model T00	1	\$2,920.00	\$2,920.00	\$32.00	\$768.00
0231	Rack Content Specify: 9117/570 - 4EIA	1	N/C	N/C		
4651	Rack Indicator, Rack #1	1	N/C	N/C		
6246	1.8m Rack Trim Kit	1	\$158.00	\$158.00		
	PDU to Wall Powercord 14', 200-240V/24A, UTG0247, PT#12	1	\$240.00	\$240.00		
6654	Power Distrib Unit Specify - Base/Side Mount, Universal					
9188	UTG0247 Connector	1	N/C	N/C		
9300	Language Group Specify - US English	1	N/C	N/C		
			Subtotal	\$3,318.00	Subtotal	\$768.00
7310-C04	HMC 1:7310-C04 Desktop Hardw.Mgmt.Console	1	\$1,830.00	\$1,830.00	\$48.00	\$1,152.00
6942-25B	7310-C04 24x7x4 WSU	1	\$192.00			\$192.00
	Hardware Management Console for POWER5 Licensed					
0961	Machine Code	1	N/C	N/C		
3631	IBM ThinkVision C170 17-inch Color Monitor	1	\$250.00	\$250.00		

6470	Power Cord (6-foot), To Wall (125V, 15A), Plug Type #4	2	\$14.00	\$28.00		
7801	Ethernet Cable, 6M, Hardware Management Console to System Unit	1	\$12.00	\$12.00		
8800	Quiet Touch Keyboard - USB, Business Black, US English, #103P	1	\$83.00	\$83.00		
8841	Mouse - Business Black with Keyboard Attachment Cable	1	\$62.00	\$62.00		
9300	Language Group Specify - US English	1	N/C	N/C		
			Subtotal	\$2,265.00	Subtotal	\$1,344.00
Storage						
1740-710	DS4000 EXP710 Storage Expansion	26	\$6,000.00	\$156,000.00	N/A	N/A
6942-25B	1740-710 24x7x4 WSU	26	\$760.00			\$19,760.00
2210	(19K1271) 2Gb Fibre Channel Short Wave GBIC	97	\$499.00	\$48,403.00		
5212	(06P5772) 2Gb FC, 36.4GB/15K Drive	338	\$1,115.00	\$376,870.00		
5601	(19K1247) 1m 50u Fiber Optic Cable (LC-LC)	52	\$79.00	\$4,108.00		
9009	Attach to the DS4500 (1742-90U)	26	N/C	N/C		
9202	Field Integrate FC EXP 710	8	N/C	N/C		
9800	(6952300) Power Cord 125V/10A, (Group 1)	26	N/C	N/C		
			Subtotal	\$587,514.00	Subtotal	\$19,760.00
1742-90U	DS4500 Midrange Disk System	3	\$49,900.00	\$149,700.00	N/A	N/A
6942-25B	1742-90U 24x7x4 WSU	3	\$1,087.00			\$3,261.00
2210	(19K1271) 2Gb Fibre Channel Short Wave GBIC	12	\$499.00	\$5,988.00		
3507	(19K1269) DS4000 2 Gbps miniHUB (LC)	4	\$899.00	\$3,596.00		
5605	(19K1248) Fiber Cable 5m Multimode (LC-LC)	6	\$129.00	\$774.00		
7111	DS4500 AIX Host Kit	1	\$4,000.00	\$4,000.00		
9202	Field Integrate DS4500	2	N/C	N/C		
9800	(6952300) Power Cord 125V/10A, (Group 1)	2	N/C	N/C		
			Subtotal	\$164,058.00	Subtotal	\$3,261.00
Software						
5773-RS3	Initial Hardw.Mgmt.Console Software Support 3 Year	1	N/A	N/A	N/A	N/A
0569	Per Processor Software Support 3 Year	1	\$675.00	N/A		\$675.00
0570	Per Processor 24x7 Software Support 3 Year	1	\$236.00	N/A		\$236.00
7000	Agreement for MCRSA	1	N/A	N/A		
			Subtotal	\$0.00	Subtotal	\$911.00
5692-A5L	System Software	1	N/C	N/C	N/A	N/A
0967	MEDIA 5765-G03 AIX 5L V5.3	1	N/C	N/C		
0968	Expansion pack	1	N/C	N/C		
0970	AIX 5L V5.3 Update CD	1	N/C	N/C		
0975	Microcode Upd Files and Disc Tool CD	1	N/C	N/C		
1004	CD-ROM Process Charge	1	\$50.00	\$50.00		
2924	English Language	1	N/C	N/C		
3410	CD-ROM	1	N/C	N/C		
			Subtotal	\$50.00	Subtotal	NA
5765-G03	AIX 5L V5.3	1	N/C	N/C	N/A	N/A
0005	Per Processor F5 AIX 5L V5.3	4	\$1,255.00	\$4,900.00		
			Subtotal	\$4,900.00	Subtotal	NA
5773-SM3	Software Maintenance for AIX, 3 Year	1	N/C	N/C	N/A	N/A
0466	F5 3 Yr SWMA for AIX per Processor Reg/Ren	4	\$1,958.00	NA		\$7,832.00
0468	F5 3 yr Services 7x24 Support per Processor Reg/Ren	4	\$496.00	NA		\$1,984.00
			Subtotal	\$0.00	Subtotal	\$9,816.00
D5A1DLL	C for AIX user Lic+SW maint 12 MO	1	\$515.00	\$515.00	N/A	N/A
E1A1FLL	C for AIX user annual SW maint renewal	2	NA	NA		\$206.00
			Subtotal	\$515.00	Subtotal	\$206.00
Client Hardware and Software						
8837E3U	Express x336, Xeon 2x3.2GHz/800MHz, 2GB, DVD-ROM	8	\$3,525.00	\$28,200.00	N/A	N/A
21P2073	Warranty Service Upgrade: 3 Yr OnSite 24x7x4 hours	8	\$450.00			\$3,600.00
30R5096	Express IBM 36.4GB 15K U320 SCSI HS Option	8	\$275.00	\$2,200.00		
93074SX	IBM S2 42U Standard Rack Cabinet	2	\$1,489.00	\$2,978.00		\$1,200.00
90P0726	IBM C170 17" CRT Monitor	1	\$175.00	\$175.00		\$90.00
25R6968	IBM Preferred Pro Full Size KB - PS/2	1	\$29.00	\$29.00		
90P0740	IBM Sleek 2-Button Mechanical Ball Mouse - PS/2	1	\$15.00	\$15.00		
			Subtotal	\$33,597.00	Subtotal	\$4,890.00

Total	\$1,302,950.00	\$69,318.00
IBM Total Configuration Discount		\$636,732.00
Total 3 Yr Cost of IBM Configuration		\$735,536.00

For additional information, please contact me directly at 770-835-6612.

Alycia E. Waller
 Technical Sales Specialist - eServer pSeries
 awaller@us.ibm.com



800.750.4239

SHOPPING CART

[▶ Your Saved Carts](#) [▶ Save This Cart](#) [▶ Edit Saved Carts](#) [▶ Send To An Associate](#)

[Continue to Checkout](#)

Quantity	Product	CDW	Usually Ships	Price	Ext. Price
3	BELKIN 16PT GIGABIT SWITCH	637031	1-2 Weeks	\$90.52	\$271.56
Click to remove an item from your cart				Sub-Total	\$271.56

Update

Clear Cart

[Continue to Checkout](#)

[Continue Shopping](#) | [Go to CDW.com Homepage](#)

Shipping Calc: