RackSaver

TPC Benchmarĸ™ C
Full Disclosure Report
for
QuatreX-64™
using
Microsoft SQL Server 2000
8.0 Enterprise Edition SP-3 - 32 bit
And
Microsoft Windows 2003
Enterprise Server - 32 bit

First Edition
Submitted:
July 8, 2003

First Version, July 8, 2003

RackSaver believes that the information included in this document is accurate as of the publication date. The information in this document is subject to change without notice. RackSaver is not responsible for any errors contained within this document. The pricing information in this FDR is accurate as of the publication date, July 8, 2003 and is generally available.

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

All performance data contained in this report were obtained in a rigorously controlled environment. Actual performance experienced by a particular customer may vary due to differences in system layout and configuration, hardware and/or software revision levels, and background system activity. The content of this document is for informational purposes only.

Table of Contents

RackSaver 1
Table of Contents 3
Abstract 4
Preface 9
Introduction 9
General Items 10
Clause 1 - Logical Database Design Related Items 13
Clause 2 - Transaction and Terminal Profiles Related Items 14
Clause 3 - Transaction and System Properties Related Items 16
Clause 4: Scaling and Database Population Related Items 18
Clause 5: Performance Metrics and Response Time Related Items 21
Clause 6: Performance Metrics and Response Time Related Items 29
Clause 7: Pricing Related Items 31
Clause 9: Audit Related Items 33
Auditor's Attestation Letter 33

Abstract

Overview

This report documents the methodology and results of the TPC Benchmark™ C test conducted on RackSaver QuatreX-64 Server. The tests were run in a client/server configuration using Eight RackSaver RS-1100's as clients. The operating system used for the benchmark was Microsoft Windows 2003 Enterprise Server on the database server and Microsoft Windows 2000 Server on the clients. The database was Microsoft SQL Server 2000 Enterprise Edition. Microsoft COM+ provided the database connection queues. All tests were done in compliance with Revision 5.0 of the Transaction Processing Council's TPC Benchmark™ C Standard Specification. Two standard TPC Benchmark™ C metrics, transactions per second (tpmC) and price per tpmC ($/tpmC) are reported and referred to in this document. The results from the tests are summarized below.

Hardware	Software	Total System Cost	tpmC	$/tpmC	Total Solution Availability Date
RackSaver QuatreX-64 Server	**Microsoft SQL Server 2000 Enterprise Edition** **Microsoft Windows 2003 Enterprise Server**	**$223,664**	**82,226.46**	**$2.72**	**Oct 21, 2003**

Auditor

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

Copies of this Full Disclosure Report can be obtained from either the Transaction Processing Performance Council or RackSaver at the following address:

Transaction Processing Performance Council (TPC)
c/o Adminstrator, TPC
Presidio of San Francisco
Bldg 572B Ruger St.
San Francisco, CA 94129-0920
Phone: (415) 561-6272, fax 415-561 6120
www.tpc.org

or

RackSaver
9449 Carroll Park Drive
San Diego, CA 92121
Phone: (858) 874-3800
www.racksaver.com

RackSaver Leader in High-Density Servers!	QuatreX-64 Server 4P	TPC-C Rev 5.1 Report Date: July 8, 2003

Total System Cost	TPC-C Throughput	Price /Performance	Availability Date
$223,664	**82,226.46**	**$2.72**	**Oct 21, 2003**

Processors	Database Manager	Operating System	Other Software	Number of Users
4 AMD Opteron Processors Model 844 (3 HyperTransport Links, 1M Cache, 1.8 GHz)	Microsoft SQL Server 2000 Enterprise Edition SP3	Microsoft Server 2003, Enterprise Server	Microsoft Windows 2000 Server w/COM+ Microsoft Visual C++ Standard	**66,000**

Clients **Server**

RackSaver QuatreX-64
w/4 AMD Opteron Processors Model 844, 3 HyperTransport Links, 1M Cache, 1.8 GHz, 32 GB RAM, 5 Compaq 5304/128 SMART Array Controllers

RackSaver RS-1132
w/2 AMD Athlon MP 2400+ Processors, 256K Cache, 2.0 GHz, 1 GB RAM

20 RackSaver Disk Pods
270 18GB 15K SCSI Drives, 24 36 GB 15K SCSI Drives

System Component	Qty	Server	Qty	Each Client
Processors	4	AMD Opteron Model 844 Processors (1.8 GHz)	2	AMD Athlon MP 2400+ Processors (2.0 GHz)
Cache		1 M		256 K
Memory	16	2 GB PC2100 DDR Reg.	2	512 MB P2100 DDR Reg.
Disk Controllers	5	Compaq 5304 SMART Array Controllers		
Disk Drives	270 24	18 GB SCSI Drives 15K 36 GB SCSI Drives 15K	1	80 GB EIDE
Total Storage		5.2 TB	1	80 GB

RackSaver	QuatreX-64 Server AMD Opteron 844 (4P, 1.8GHz) with 4 RackSaver RS-1132	TPC-C REV 5.1 Report Date: 8-July-2003

Description	Part Number	Third Party Brand	Pricing	Unit Price	Qty	Extended Price	3 yr. Maint. Price
Server Hardware							
Server RS872Q64S Bundle (includes the following)	**RS872Q64S**		1	34,950	1	34,950	3,500
Quatrex-64 Base System	BBAMQUATREX64				1	0	
AMD Opteron Processor Model 844 (1.8 GHz)	CPAMOP8441.8				4	0	
1.44, 3 1/2 Black	SONY1.44BLACK				1	0	
DDR 2GB PC2100 ECC Reg	MEGED2GBER				16	0	
IDE 40GB 7200 8MB	HDWE4018M72				1	0	
Slimline 24X CDROM	24X1UCD-PANAS				1	0	
Compaq 5304 SMART Array Controller	COCO158939-B21	Compaq/HP	1	995	5	4,975	
DDS4-20/40 Tape Drive	SDT11000/BM		1	800	1	800	
17" Monitor 1280x1024	MOKD17		1	150	1	150	
104 Windows Keyboard	7603		1	8	1	8	
3-button PS/2 Mouse	MILO3BUTTON		1	7	1	7	
Belkin 8 Port Omniview PRO2	CBBEF1DA108T	Belkin	1	250	1	250	
3 Year Warranty, 4 Hour Response Time	EWSSQUATREX3YR4HR				1	0	
					Subtotal	41,140	3,500
Server Storage							
RackStor 4150-250 (includes the following)	**RS4150-250**		1	4,350	18	78,300	Incl.
4U SCA RAID Chassis/400W black	RMC4D2-6-0S				18	0	
SCSI 18GB 15K 8MB 80P	HDSE18SAM1580				270	0	
SCSI Cable/Internal Daisy Chain	HDC-23C04				36	0	
SCSI Cable/Int 68 pin W/EXT con	HDC-23C06				18	0	
Ext SCSI Cable 2M 68pin 2 VHDCI	CBRA1402-006				18	0	
SCSI Male Terminator	SCA-TERM160M				18	0	
3 Year Warranty, 4 Hour Response Time					18	0	
RackStor 4150-400 (Includes the following)	**RS4150-400**		1	5,250	2	10,500	Incl.
4U SCA RAID Chassis/400W black	RMC4D2-6-0S				2	0	
SCSI 36GB 15K 8MB 80P	HDSE36S8M1580				24	0	
SCSI Cable/Internal Daisy Chain	HDC-23C04				4	0	
SCSI Cable/Int 68 pin W/EXT con	HDC-23C06				2	0	
Ext SCSI Cable 2M 68pin 2 VHDCI	CBRA1402-006				2	0	
SCSI Male Terminator	SCA-TERM160M				2	0	
3 Year Warranty, 4 Hour Response Time					2	0	
UPS 2200VA RM-3U*	UPAPSU2200RM3U		1	900	4	3,600	0
					Subtotal	92,400	0
Server Software							
Microsoft SQL Server 2000, Enterprise Edition	810-00846	Microsoft	2	16,541	4	66,164	5,850
Microsoft Visual C++ Standard	254-00170	Microsoft	2	109	1	109	Inc Above
Microsoft Server 2003, Enterprise Server	P72-00264	Microsoft	2	2,399	1	2,399	Inc Above
					Subtotal	68,672	5,850

Client Hardware							
Client Hardware (includes the following)	**RS-1132**		1	1,650	4	6,600	1000
RackSaver RS-1100 Chassis	CHRA1100				4	0	0
Tyan S2469GN IDE AMD MP	MBTYS2469GN				4	0	0
1.44, 3 1/2 Black	SONY1.44BLACK				4	0	0
Slimline 24X CDROM	24X1UCD-PANAS				4	0	0
RackSaver PS 1U 300W SPI	RSP-SP1300				4	0	0
AMD Athlon MP Processor 2400+ (2.0 GHz)	CPAMMP2400				8	0	0
IDE 80GB 7200 8MB	HDWE8018M72				4	0	0
DDR 512MB PC2100 ECC Reg	MECODR21512ER				8	0	0
Copper 1U HeatSink/Fan	SRF-1UCOPPER-2				8	0	0
3 Year Warranty, 4 Hour Response Time					4	0	
				Subtotal		6,600	1,000
Client Software							
Microsoft Windows 2000 Server	C11-00821	Microsoft	2	738	4	2,952	Inc. Above
				Subtotal		2,952	0
Miscellaneous							
Linksys 10/100 Switch*	CD7G9L	Linksys	1	50	3	150	0
52U rack 4 post	RCK4P52U		1	700	2	1400	0
				Subtotal		1,550	0
				Other Discounts*			
				Total		**$213,314**	**$10,350**

Notes:

* 10% or minimum 2 spares added in place of on-site service

Pricing: 1=RackSaver, 2=Microsoft

Audited by Performance Metrics

Three Year Cost of Ownership: $223,664

tpmC Rating: 82,226.46

$ / tpmC: 2.72

MQTh, computed Maximum Qualified Throughput 82,226.46 tpmC

Response Times (in seconds)	Average	90 th	Max
- New order	0.51	1.09	28.32
- Payment	0.43	1.01	55.70
- Order Status	0.45	1.02	13.65
- Delivery (interactive)	0.10	0.11	1.11
- Delivery (deferred)	7.76	16.60	43.20
- Stock-Level	0.96	1.62	21.18
- Menu	0.10	0.11	63.10

Response time delay added for emulated components

Menu 0.1
Resp. 0.1

Transaction Mix, in percent of total transactions

- New-Order	44.90 %
- Payment	43.03 %
- Order-Status	4.03 %
- Delivery	4.02 %
- Stock-Level	4.02 %

Keying/Think Times (in seconds),

	Min		Average		Max	
- New-Order	18.02	0.0	18.03	12.05	18.12	120.41
- Payment	3.01	0.0	3.03	12.04	3.11	120.41
- Order-Status	2.01	0.0	2.03	10.02	2.10	100.41
- Delivery	2.01	0.0	2.03	5.05	2.10	50.41
- Stock-Level	2.01	0.0	2.03	5.03	2.07	50.41

Test Duration

- Ramp-up time	31 minutes
- Measurement interval	120 minutes
- Number of checkpoints	4
- Checkpoint interval	29.9 minutes
- Number of transactions (all types) Completed in measurement interval	22,859,999

Preface

The Transaction Processing Performance Council (TPC) developed The TPC Benchmark™ C. The TPC was founded to define transaction processing benchmarks and to disseminate objective, verifiable performance data to the industry. This full disclosure report is based on the TPC Benchmark C Standard Specification Version 5.1.

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

Introduction

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

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

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

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

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

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

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

General Items

Benchmark Sponsor

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

RackSaver was the benchmark sponsor for this TPC Benchmark™ C.

Application Code and Definition Statements

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

Appendix A contains all source code implemented in this benchmark.

Parameter Settings

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

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

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

Configuration Diagrams

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

The following pages contain the diagrams for both the tested and priced configurations.

Figure 1.1: Measured Configuration:

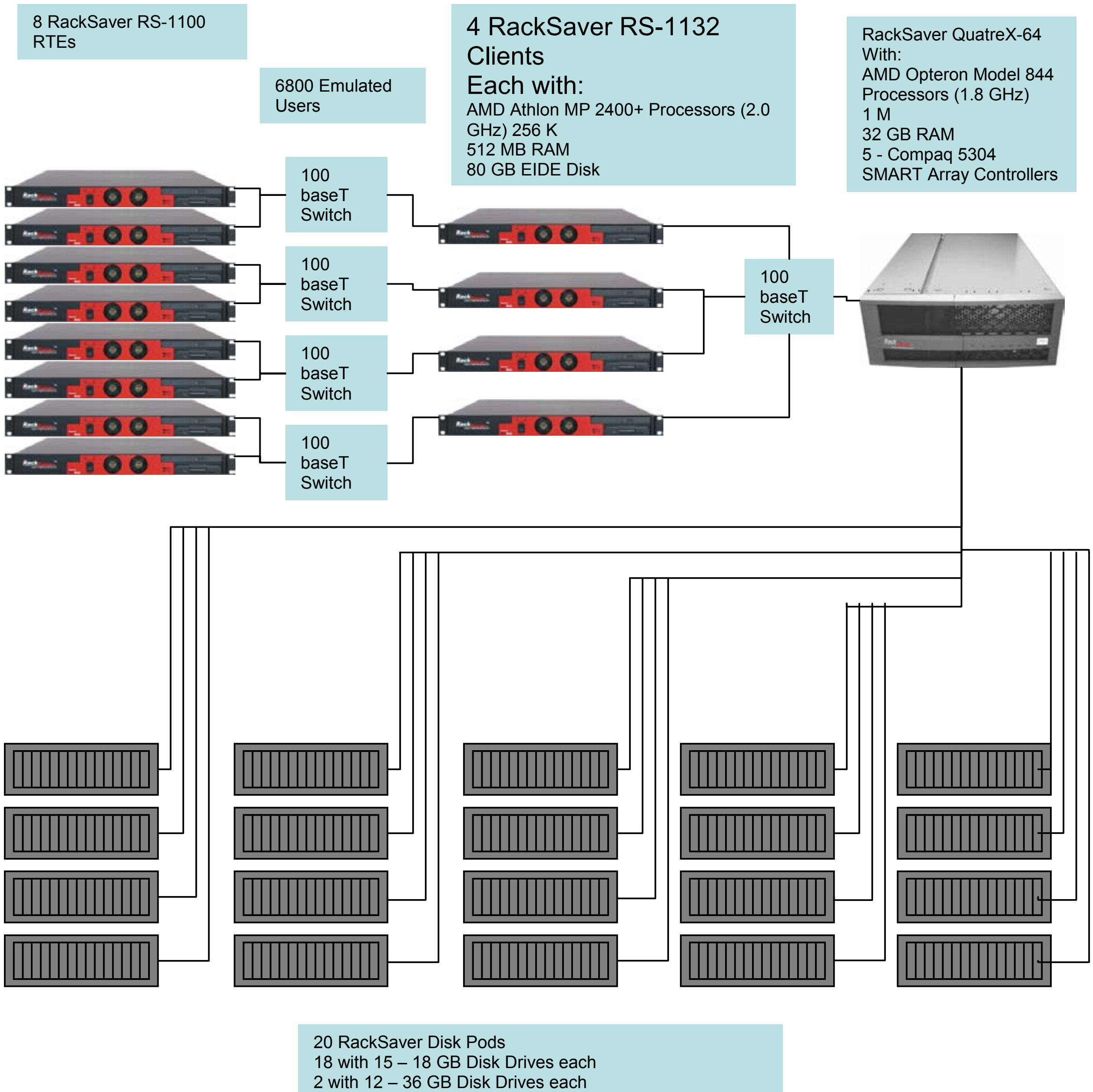
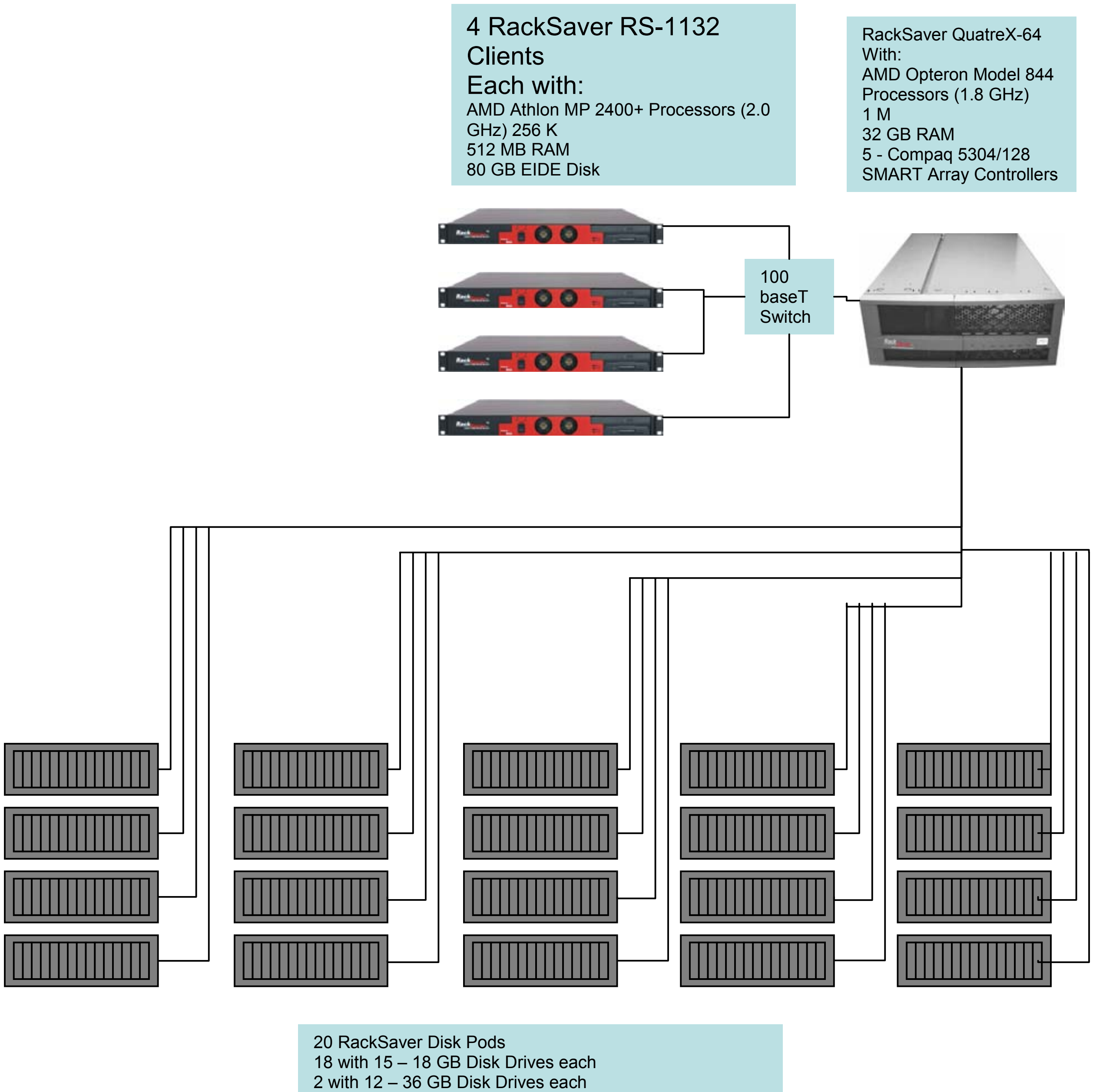


Figure 1.2: Priced Configuration:



<u>Clause 1 - Logical Database Design Related Items</u>

Table Definitions

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

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

Physical Organization of the Database

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

The tested database configuration used 180 disk drives. The physical organization is documented in Table 5: Data Distribution.

Insert and Delete Operations

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

Insert and delete functions were fully operational during the running of the benchmark.

Horizontal or Vertical Partitioning

While there are few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC-C benchmark (see Clause 1.6), any such partitioning must be disclosed.

Partitioning was not used in this benchmark.

Replication

Replication tables, if used, must be disclosed (see Clause 1.4.6).

Replication was not used in this benchmark.

Table Attributes

Additional and/or duplicated attributes in any table must be disclosed, along with a statement on the impact on performance (see Clause 1.4.7).

No additional attributes were used in this benchmark.

Clause 2 - Transaction and Terminal Profiles Related Items

Random Number Generation

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

The random number generation was handled internally in the Microsoft BenchCraft RTE program. The independent auditing process verified this.

Screen Layout

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

All screen layouts followed the Standard Specifications.

Terminal Verification

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

The auditor with a thorough execution of the five transaction types, using Microsoft Internet Explorer, verified the terminal features.

Intelligent Terminals

Any usage of presentation managers or intelligent terminals must be explained.
Comment 1: *The intent of this clause is to describe any special manipulations performed by a local terminal or workstation to off-load work from the SUT. This includes, but is not limited to: screen presentations, message bundling, and local storage of TPC-C rows.*
Comment 2: *This disclosure also requires that all data manipulation functions performed by the local terminal to provide navigational aids for transaction(s) must also be described. Within this disclosure, the purpose of such additional function(s) must be explained.*

The application code responsible for processing the data was executed on the clients. HTML Screen manipulation commands were downloaded to the web browser, which controlled the input and output graphics. This code is documented in Appendix A. IIS (Microsoft Internet Information Server) was involved in processing and presenting this data.

Transaction Profiles

The percentage of home and remote order-lines in the New-Order transactions must be disclosed.
The percentage of New-Order transactions that were rolled back as a result of an unused item number must be disclosed.
The number of items per orders entered by New-Order transactions must be disclosed.
The percentage of home and remote Payment transactions must be disclosed.
The percentage of Payment and Order-Status transactions that used non-primary key (C_LAST) access to the

database must be disclosed.
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: Transaction Statistics

Transaction	Function	Value
New Order	**Home Warehouse Order Lines**	99%
	Remote Warehouse Order Lines	1%
	Rolled Back Transactions	1%
	Average Lineitems Per Order	10
Payment	**Home Warehouse Transactions**	85%
	Remote Warehouse Transactions	15.01%
	Non-Primary Key Access	59.98%
Order Status	**Non-Primary Key Access**	60.04%
Delivery	**Delivery Transactions Skipped**	0.0

Transaction Mix

The mix (i.e., percentages) of transaction types seen by the SUT must be disclosed. (8.1.3.11)

Table 2.2: Transaction Mix

Transaction	Percentage
New Order	44.90%
Payment	43.03%
Delivery	4.02%
Stock Level	4.02%
Order Status	4.03%

Deferred Delivery Mechanism

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

The application creates a semaphore-base thread pool consisting of a user-specified number of threads, which open ODBC connections on the database. When a Delivery transaction is posted one of these threads makes the database call while the transaction's original thread returns control to the user. Upon completion the Delivery thread writes an entry in the Delivery log and returns to the thread pool.

The source code is listed in Appendix A.

Clause 3 - Transaction and System Properties Related Items

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

All ACID tests were conducted successfully according to specification.

Atomicity

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

Completed Transaction

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

Aborted Transaction

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

Consistency

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

Consistency conditions one through four were tested using a script to issue queries to the database. The results of the queries verified that the database was consistent for all four tests. A run was executed under full load lasting over two hours and including several checkpoints. The script was re-executed and the result of the same queries verified that the database remained consistent after the run.

Isolation

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

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

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

For Isolation test seven, case A was followed.

Durability

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

Loss of Data

Loss of data was verified on a 200 Warehouse database. The RTEs were used to generate the transaction load of 2000 users for the test. To demonstrate recovery from a permanent failure of durable media containing TPC-C tables, the following steps were executed:

1) A 720 Warehouse database was generated with similar characteristics to the large database.
2) The database was backed up using SQL scripts.
3) A sum of D_NEXT_O_ID was recorded.
4) 7200 users were started via the RTEs.
5) The system was run in steady state for 5 minutes.
6) A checkpoint was completed and execution was continued for a additional minute.
7) Two disk drives in the data array were removed causing SQL Server errors.
8) The RTE was paused and allowed to finish processing.
9) The RTE was stopped.
10) SQL Server was stopped and restarted and a dump of the transaction log was saved.
11) SQL Server was stopped and the system was shutdown.
12) The failed disks were swapped.
13) The machine was booted and SQL Server was started.
14) The TPC-C database was dropped and restored from backup with norecovery.
15) The transaction log was restored and the database was allowed to recover.
16) A new count of D_NEXT_O_ID was taken.
17) This number was compared with the number of new orders reported by the RTE.
18) Consistency test #3 was executed and verified.

Combined Loss of System Test (Instantaneous Interruption and Loss of Memory) and Loss of Log

Loss of System Test and Loss of Log were performed on the full database with 6,800 warehouses in a combined test. The RTEs were used to generate a transaction load of 68,000 users for the test. To validate system recovery an instantaneous interruption was caused by removing power to the Server, the following steps were executed:

1) A sum of D_NEXT_O_ID was taken.
2) 68,000 users were started via the RTEs.
3) The system was run is steady state for 5 minutes
4) One transaction log disk drive was removed with no effect on the database server operations.
5) The system ran for an additional 5 minutes.
6) A checkpoint was completed and execution was continued for 30 seconds.
7) Power was removed from the server causing instantaneous interruption.
8) The RTE’s were paused and allowed to complete transactions received via the clients.
9) The RTE’s were stopped.
10) Power was reconnected and the system was rebooted.
11) SQL Server was started and recovery completed.
12) A new count of D_NEXT_O_ID was taken.
13) This number was validated against the calculated number reported by the RTEs.

Clause 4: Scaling and Database Population Related Items

Cardinality of the Tables

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

The database was generated with 7,200 warehouses, and the audited performance run used 6,800 warehouses.

Table 4.1: Table Cardinality

Table	Initial Cardinality
Warehouse	7,200
District	72,000
Customer	216,000,000
New Order	216,000,000
Orders	216,000,000
History	216,000,000
Order Line	2,147,483,647
Item	100,000
Stock	720,000,000
Deleted Warehouses	0

Distribution of Database Tables and Logs

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

The system was configured with 270 18GB 15k SCSI disks to be used by the database. There were 270 data drives connected to 5 Compaq SMART Array Controllers configured as RAID 0 (60 each on 4 controllers and 30 on the remaining controller). Most logical data drive contained 3 partitions for miscellaneous, customer and stock, and backup data. Raw file systems were used except for the NTFS formatted backup partitions. The Log Drives used 24 36GB 15k disks and were mirrored with RAID 1/0. The configuration is further detailed below in Table 4.2.

Table 4.2: Data Distribution

Controller	DB Components	Partition	Size	Disks
0	Miscellaneous Customer and Stock Backup	F: M:	43,620 MB 80,340 MB 903 GB	60 – 18 GB 15K
1	Miscellaneous Customer and Stock Backup	G: N:	56,460 MB 103,980 MB 867 GB	60 – 18 GB 15K
2	Miscellaneous Customer and Stock Backup	H: O: T:	56,460 MB 34,816 MB 867 GB	60 – 18 GB 15K
3	Miscellaneous Customer and Stock Backup	I: P: U:	43,620 MB 80,340 MB 903 GB	60 – 18 GB 15K
4 (Array A)	Transaction Log	E:	204,720 MB	24 – 36 GB 15K
4 (Array B)	Miscellaneous Customer and Stock	J: Q:	30,780 MB 56,700 MB	30 – 18 GB 15K

Database Model

A statement must be provided that describes:

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

Microsoft SQL Server 2000 Enterprise Edition is a relational DBMS.

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

Mapping Partitions/Replications

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

The database was not replicated.

60-Day Space

Details of the 60-day space computations along with proof that the database is configured to sustain 8 hours of growth for the dynamic tables (Order, Order-Line, and History) must be disclosed (see Clause 4.2.3).

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

1. The current log space was recorded by running *dbcc sqlperf(logspace)*
2. Transactions were run against the database with a full user load.
3. The final log space usage was recorded by running *dbcc sqlperf(logspace)*
4. The space used was calculated as the difference between the first and second queries.
5. The number of NEW-ORDERS was retrieved from the RTE report generated for the entire run.
6. The total space used was divided by the number of NEW-ORDERS producing a size per NEW-ORDER.

7. The NEW-ORDER size was multiplied by the measured tpmC rate and multiplied by 480 minutes.

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

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

Clause 5: Performance Metrics and Response Time Related Items

Measured tpmC

Measured tpmC must be reported.

Measured tpmC: 82.226.46 tpmC
Price per tpmC: $2.72 per tpmC

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: Response Times

Transaction	Average	Maximum	90 %
New Order	0.51	28.32	1.09
Payment	0.43	55.70	1.01
Delivery	0.10	1.11	0.11
Stock Level	0.96	21.18	1.62
Order Status	0.45	13.65	1.02
Deferred Delivery	7.76	43.20	16.60
Menu	0.10	63.10	0.11

Keying and Think Times

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

Table 5.2: Keying Times

Transaction	Average	Minimum	Maximum
New Order	18.03	18.02	18.12
Payment	3.03	3.01	3.11
Delivery	2.03	2.01	2.10
Stock Level	2.03	2.01	2.07
Order Status	2.03	2.01	2.10

Table 5.3: Think Times

Transaction	Average	Minimum	Maximum
New Order	12.05	0.00	120.41
Payment	12.04	0.00	120.41
Delivery	5.05	0.00	50.41
Stock Level	5.03	0.00	50.41
Order Status	10.02	0.00	100.41

Response Time Distribution Curves

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

Figure 5.1 - New-Order Transaction Response Time Distribution

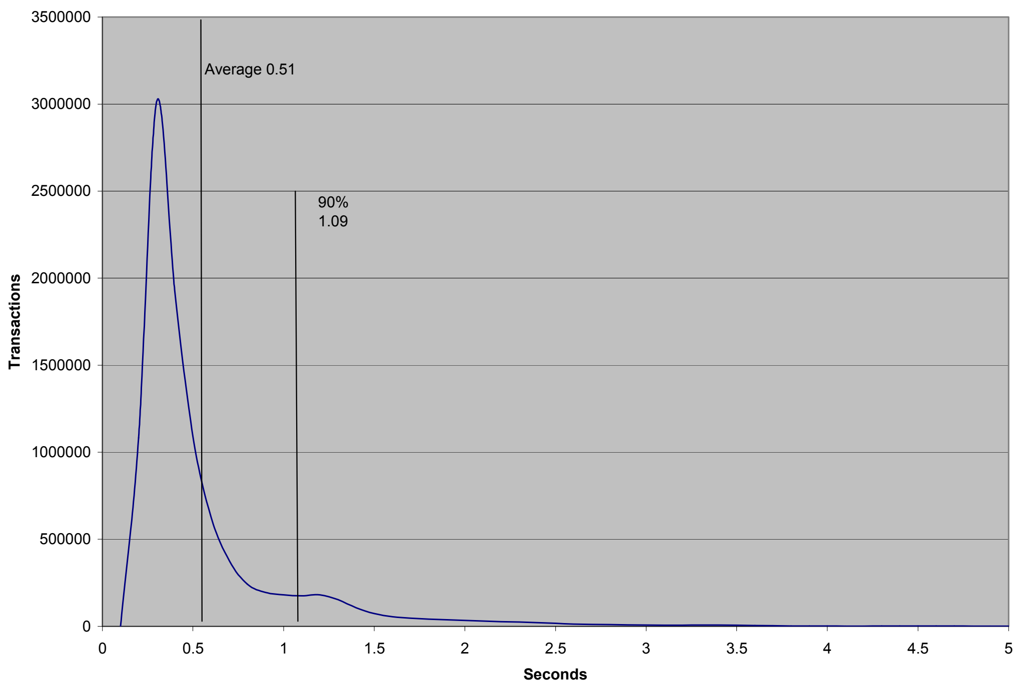


Figure 5.2 - Payment Transaction Response Time Distribution

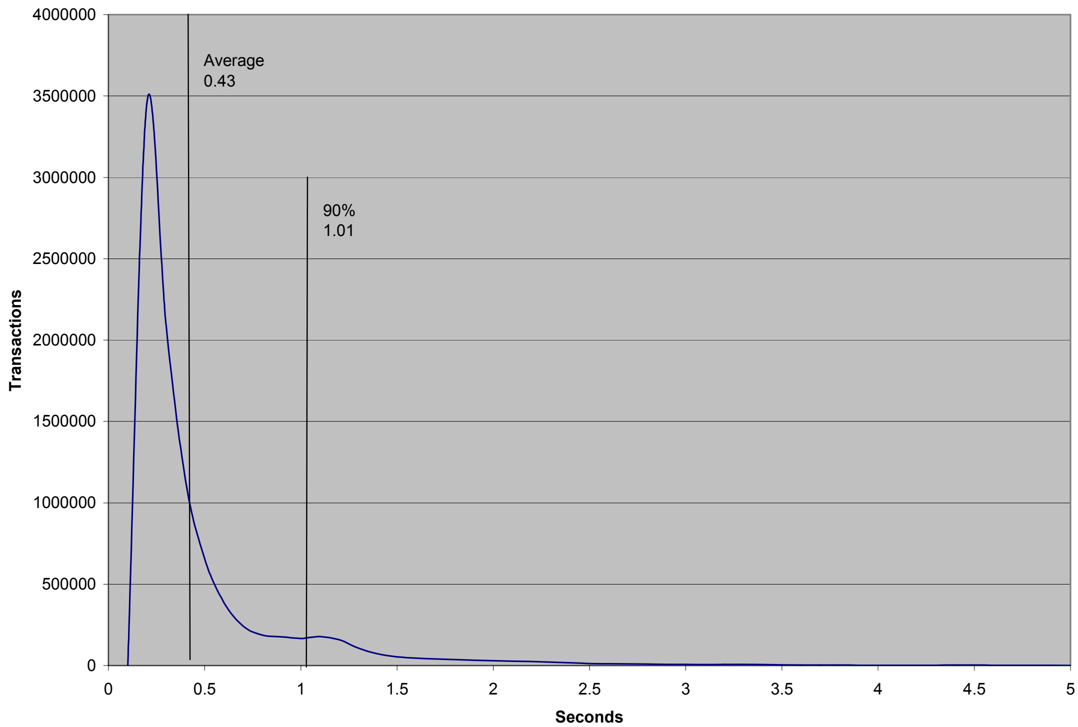


Figure 5.3 – Stock Level Transaction Response Time Distribution

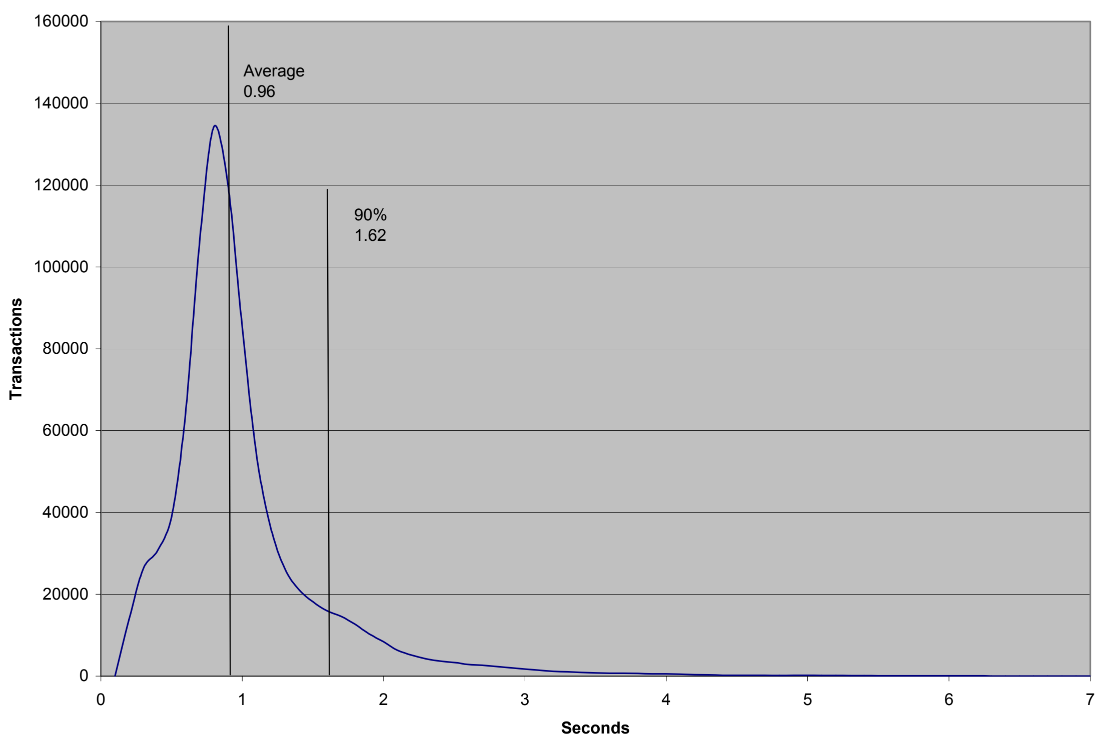


Figure 5.4 – Order Status Transaction Response Time Distribution

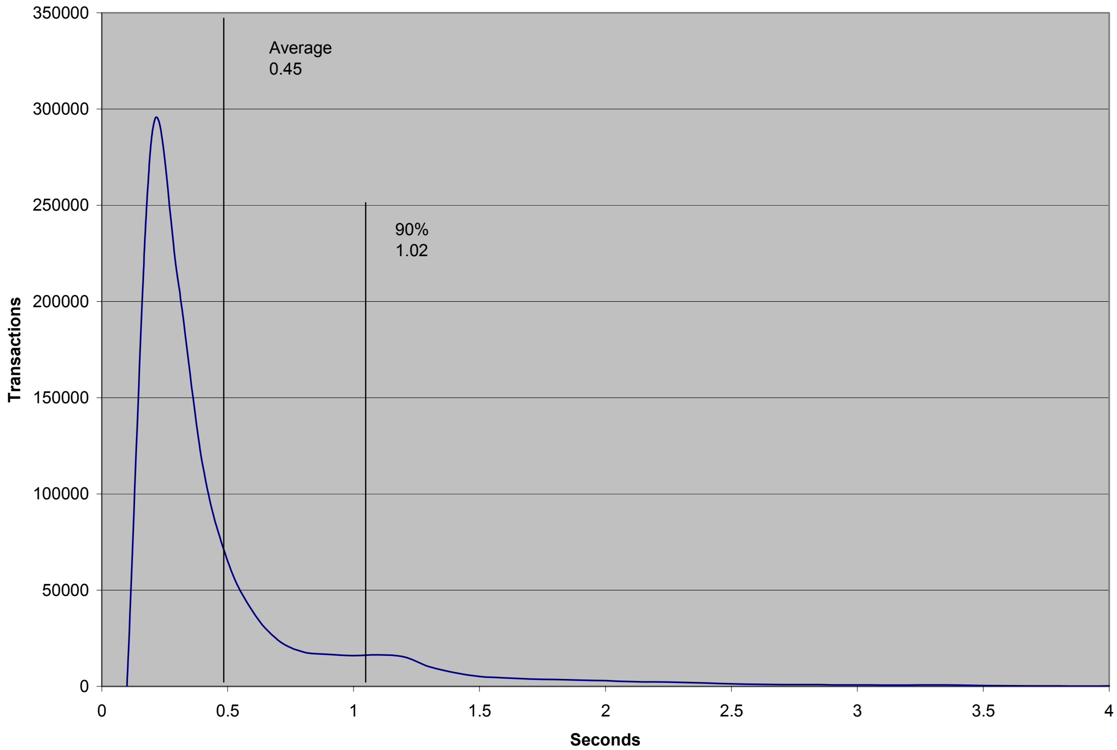
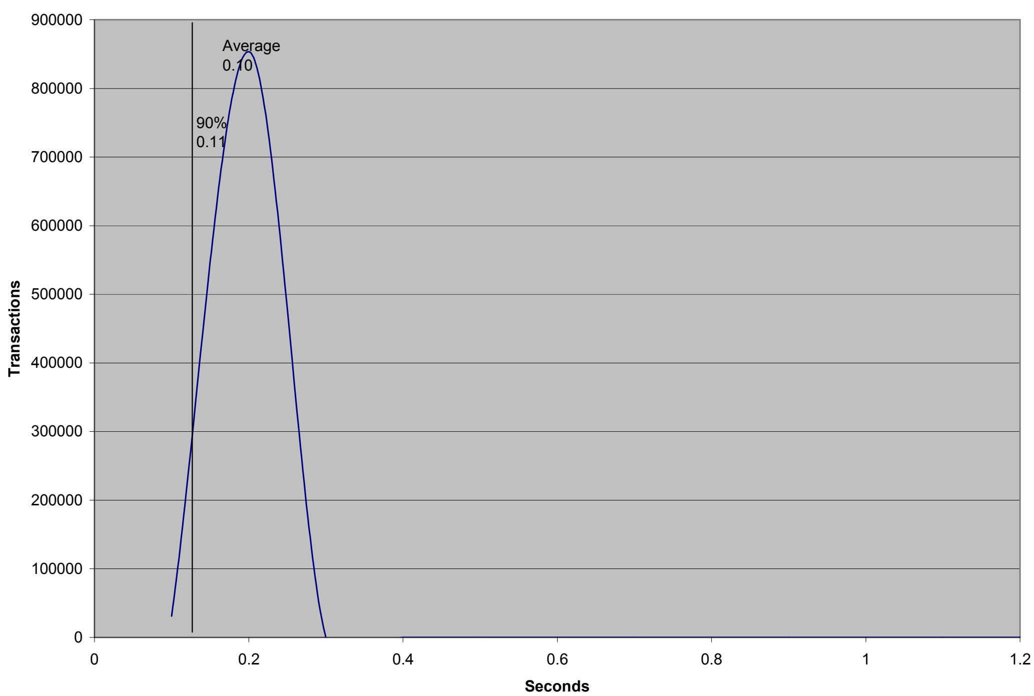


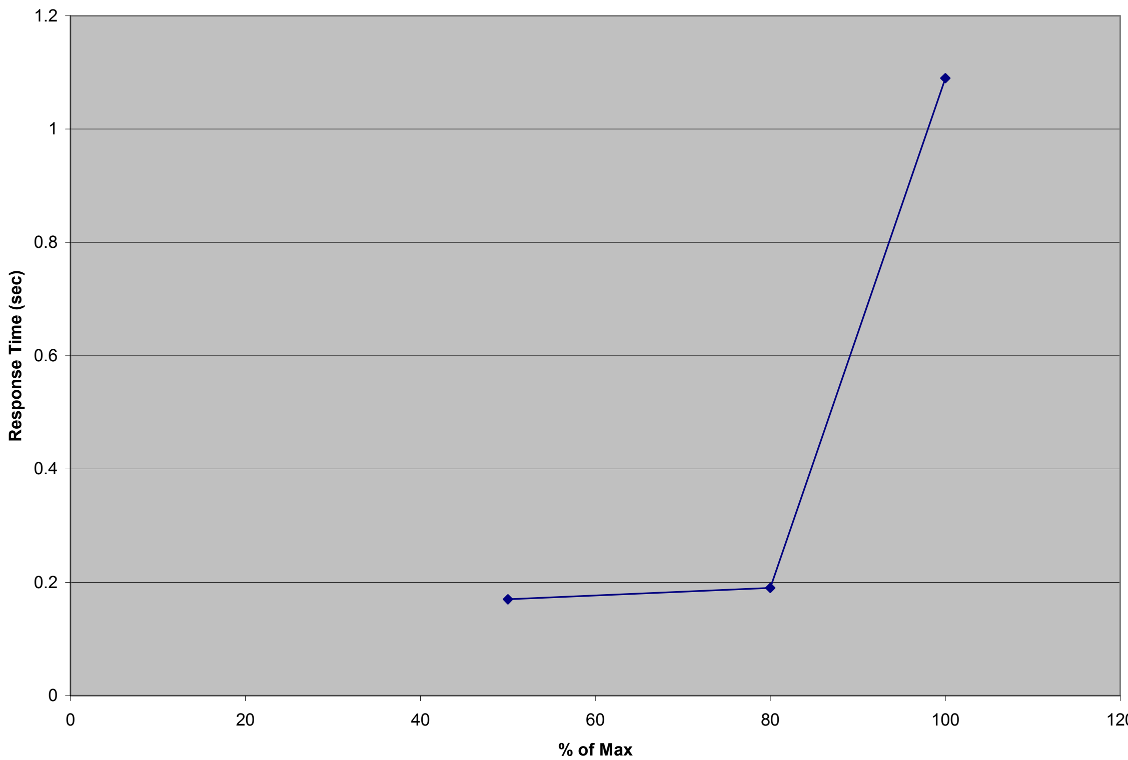
Figure 5.5 – Delivery Transaction Response Time Distribution



New Order Response Time vs. Throughput Performance

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

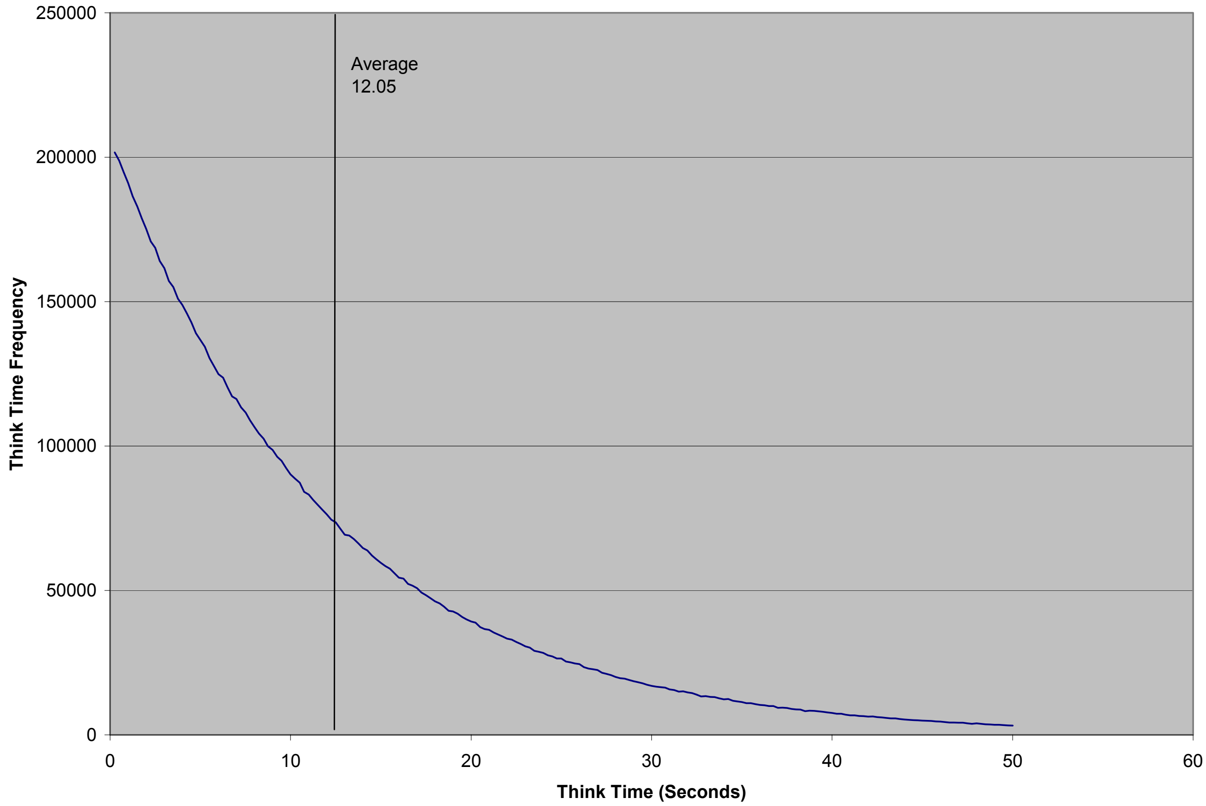
Figure 5.6 – New Order Response Time vs. Throughput



New Order Think Time Distribution

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for the New-Order transaction.

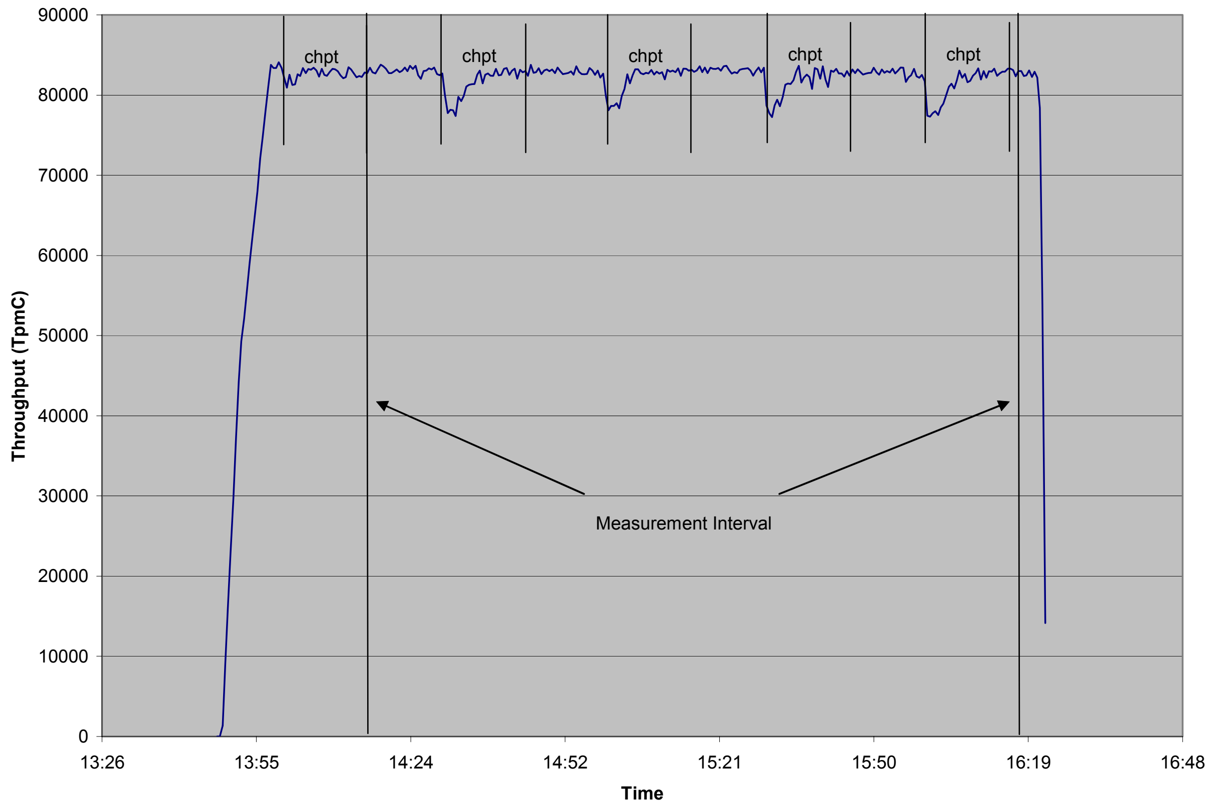
Figure 5.7 – New Order Think Time Distribution



New Order Throughput vs. Elapsed Time

A graph of throughput versus elapsed time (see Clause 5.6.4) must be reported for the New-Order transaction

Figure 5.8 – New Order Throughput vs. Time



Steady State Methodology

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

By using the monitoring tools on the RTE, a steady state was determined. Figure 5.8 further supports the level chosen by the utilities used.

Work Performed during Steady State

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

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

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

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

The RTE transmissions were sent to application processes running on the client machines through Ethernet LANs. These client application processes handled all screen I/O as well as all requests to the database on the server. The applications communicated with the database server over the Ethernet LAN using ODBC and RPC calls.
To perform checkpoints at specific intervals, we set SQL Server *recovery interval* to 300 and wrote a script to schedule multiple checkpoints at specific intervals. The script included a wait time between each checkpoint equal of 29.9 minutes so that the checkpoint interval was an integral multiple of the measurement interval, which was 120 minutes. The checkpoint script was started manually after the RTE had all users logged in and the database had achieved steady state.

At each checkpoint, Microsoft SQL Server wrote to disk all memory pages that had been updated but not yet physically written to disk. The positioning of the measurement interval was verified to be clear of the guard zones and is depicted on the graph in Figure 8.

Measurement Interval

A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included. (8.1.6.12)
The measurement interval was 7200 minutes.

Measurement Period Duration and Checkpoint Duration

The start time and duration in seconds of at least the four (4) longest checkpoints during the measurement interval must be disclosed (see clause 5.5.2.2(2)) (8.1.6.11)
A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included. (8.1.6.12)

	Start	**End**	**Duration**
Measurement Interval	14:16:32	16:16:32	7,200
Checkpoint 1	14:30:27	14:45:27	900
Checkpoint 2	15:00:22	15:15:22	900
Checkpoint 3	15:30:17	15:45:17	900
Checkpoint 4	16:00:12	16:15:13	901

Clause 6: Performance Metrics and Response Time Related Items

RTE Parameters
The RTE input parameters, code fragments, functions, etc. used to generate each transaction input field must be disclosed. (8.1.7.1)
Comment: *The intent is to demonstrate the RTE was configured to generate transaction input data as specified in Clause 2.*

The RTE input parameters are listed in Appendix C - Tunable Parameters.

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. The results of the test described in Clause 6.6.3.4 must be disclosed. (8.1.7.2)

No components were emulated.

Benchmarked and Targeted System Configuration Diagrams
A complete functional diagram of both the benchmark configuration and the configuration of the proposed (target) system must be disclosed. A detailed list of all software and hardware functionality being performed on the Driver System, and its interface to the SUT must be disclosed (see Clause 6.6.3.6). (8.1.7.3)

The driver system performed transaction data generation and communication to the client through the standard web browser (HTTP) protocol. It also captured and timestamped the SUT output data for post-processing of the reported metrics. No other functionality was included on the driver system.

Figures 1.1 & 1.2 of this report contain detailed diagrams of both the benchmark configuration and the priced configuration.

Network Configuration
The network configurations of both the tested services and the proposed (target) services which are being represented and a thorough explanation of exactly which parts of the proposed configuration are being replaced with the Driver System must be disclosed (see Clause 6.6.4). (8.1.7.4)

The network configurations of the benchmarked and priced configurations were identical.

Network Bandwidth
The bandwidth of the network(s) used in the tested/priced configuration must be disclosed. (8.1.7.5)

The bandwidth of the tested and priced networks were as follows:

- 10 BaseT (10 Mbit/sec) network segments between the RTE/Emulated Users and the switch.
- 100 BaseT (100 Mbit/sec) between the Clients and Server.

Operator Intervention
If the configuration requires operator intervention (see Clause 6.6.6), the mechanism and the frequency of this intervention must be disclosed. (8.1.7.6)

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

Clause 7: Pricing Related Items

Hardware and Software List

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

Pricing source(s) and effective date(s) of price(s) must also be reported. (8.1.8.1)

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. (8.1.8.2)

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

Availability Date

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. (8.1.8.3)

Hardware Availability Date: Oct 21, 2003
Software Availability Date: Oct 21, 2003

Measured TpmC

A statement of the measured tpmC, as well as the respective calculations for 5-year pricing, price/performance (price/tpmC), and the availability date must be included. (8.1.8.4)

Maximum Qualified Throughput: 82,226.46 tpmC
Price Performance Metric: $2.72

Country Specific Pricing

Additional Clause 7 related items may be included in the Full Disclosure Report for each country specific priced configuration. Country specific pricing is subject to Clause 7.1.7. (8.1.8.5)

This system is priced for the United States of America.

Usage Pricing

For any usage pricing, the sponsor must disclose (8.1.8.6):
• Usage level at which the component was priced.
• A statement of the company policy allowing such pricing.

***Comment**: Usage pricing may include, but is not limited to, the operating system and database management software.*

The component pricing based on usage is shown below:

- 4 Microsoft Windows 2000 Server Licenses
- 1 Microsoft Windows 2003 Enterprise Server License
- 1 Microsoft SQL Server 2000 Enterprise Edition License.
- 1 Microsoft Visual C++ 32 bit Edition
- 3 Year Support for Hardware Components.

System Pricing

System pricing should include subtotals for the following components: Server Hardware, Server Software, Client Hardware, Client Software, and Network Components used for terminal connection (see Clause 7.2.2.3). Clause 6.1 describes the Server and Client components. An example of the standard pricing sheet is shown in Appendix B. (8.1.8.7)

System pricing must include line item indication where non-sponsoring companies' brands are used. System pricing must also include line item indication of third party pricing. See example in Appendix B. (8.1.8.8)

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

Clause 9: Audit Related Items

Auditor
The auditor's name, address, phone number, and a copy of the auditor's attestation letter indicating compliance must be included in the Full Disclosure Report. (8.1.9.1)

A review of the pricing model is required to ensure that all components required are priced (see Clause 9.2.8). The auditor is not required to review the final Full Disclosure Report or the final pricing prior to issuing the attestations letter. (8.1.9.2)

This TPC-C benchmark has been audited by Lorna Livingtree of Performance Metrics.

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

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

Transaction Processing Performance Council
c/o Adminstrator, TPC
Presidio of San Francisco
Bldg 572B Ruger St.
San Francisco, CA 94129-0920
Phone: (415) 561-6272, fax 415-561 6120
www.tpc.org

or:

RackSaver
9449 Carroll Park Drive
San Diego, CA 92121
Phone: (858) 874-3800
www.racksaver.com

Auditor's Attestation Letter
On following Page

Appendix A:
Source Code

Error.h

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

#pragma once

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

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

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


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


#define ERR_TYPE_LOGIC
				-1
	//logic error in program;
internal error
#define ERR_SUCCESS
					0
		//success (a non-error
error)
#define ERR_BAD_ITEM_ID
					1
		//expected abort record in
txnRecord
#define ERR_TYPE_DELIVERY_POST
				2
	//expected delivery post failed
#define ERR_TYPE_WEBDLL
					3
		//tpcc web generated error
#define ERR_TYPE_SQL
				4
	//sql server generated error
#define ERR_TYPE_DBLIB
				5
	//dblib generated error
#define ERR_TYPE_ODBC
				6
	//odbc generated error
#define ERR_TYPE_SOCKET
					7
		//error on communication
socket client rte only
#define ERR_TYPE_DEADLOCK
				8
	//dblib and odbc only deadlock
condition
#define ERR_TYPE_COM
				9
	//error from COM call
#define ERR_TYPE_TUXEDO
					10
		//tuxedo error
#define ERR_TYPE_OS
					11
		//operating system error
#define ERR_TYPE_MEMORY
					12
		//memory allocation error
#define ERR_TYPE_TPCC_ODBC
				13
	//error from tpcc odbc txn module
#define ERR_TYPE_TPCC_DBLIB
				14
	//error from tpcc dblib txn
module
#define ERR_TYPE_DELISRV
				15
	//delivery server error
#define ERR_TYPE_TXNLOG
					16
		//txn log error
#define ERR_TYPE_BCCONN
					17
		//Benchcraft connection
class
#define ERR_TYPE_TPCC_CONN
				18
	//Benchcraft connection class
#define ERR_TYPE_ENCINA
					19
		//Encina error
#define ERR_TYPE_COMPONENT
				20
	//error from COM component
#define ERR_TYPE_RTE
				21
	//Benchcraft rte
#define ERR_TYPE_AUTOMATION
				22
	//Benchcraft automation errors
```

Appendix A:
Source Code

```
#define ERR_TYPE_DRIVER
                                        23
                //Driver engine errors
#define ERR_TYPE_RTE_BASE
                                24
        //Framework errors
#define ERR_BUF_OVERFLOW
                                25
        //Buffer overflow during receive
// TPC-W error types
#define ERR_TYPE_TPCW_CONN
                                50
        //Benchcraft connection class
#define ERR_TYPE_TPCW_HTML
                                51
        //error from TpcwHtml dll
#define ERR_TYPE_TPCW_USER
                                52
        //error from TPC-W user class
#define ERR_TYPE_TPCW_ENG_BASE
                                53
#define ERR_TYPE_TPCW_ENG_OS
                        54
#define ERR_TYPE_HTML_RESP
                                55
#define ERR_TYPE_TPCW_ODBC
                                56
#define ERR_TYPE_SCHANNEL
                                57

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



class CBaseErr
{
public:

        CBaseErr(LPCTSTR szLoc = NULL)
        {
                m_idMsg         =
INV_ERROR_CODE;


                if (szLoc)
                {
                        m_szLoc = new
char[m_szLoc_size];
                        strcpy(m_szLoc,
szLoc);
                }
                else
                        m_szLoc = NULL;

                m_szApp         = new
char[m_szApp_size];

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

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


                if (szLoc)
                {
                        m_szLoc = new
char[m_szLoc_size];
                        strcpy(m_szLoc,
szLoc);
                }
                else
                        m_szLoc = NULL;

                m_szApp         = new
char[m_szApp_size];

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

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

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

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

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

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

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

```
	virtual int ErrorType() = 0;  //
a value which distinguishes the kind of
error that occurred
	virtual char *ErrorText() = 0; //
a string (i.e., human readable)
representation of the error


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

	//short	m_errType;
};

class CSocketErr : public CBaseErr
{
public:
	enum Action
	{
		eNone = 0,
		eSend,
		eSocket,
		eBind,
		eConnect,
		eListen,
		eHost,
		eRecv,
		eGetHostByName,
		eWSACreateEvent,
		eWSASend,
		eWSASendImage,
		eWSAGetOverlappedResult,
		eWSARecv,
		eWSARecvImage,
		eWSAWaitForMultipleEvents,
		eWSAStartup,
		eWSAResetEvent,
		eNonRetryable,
	};

	CSocketErr(Action eAction,
LPCTSTR szLocation = NULL);

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

	Action	m_eAction;
	char	*m_szErrorText;

	int ErrorType() { return
ERR_TYPE_SOCKET;};
	char *ErrorText(void);

};

class CSystemErr : public CBaseErr
{
public:
	enum Action
	{
		eNone = 0,
		eTransactNamedPipe,
		eWaitNamedPipe,
		eSetNamedPipeHandleState,
		eCreateFile,
		eCreateProcess,
		eCallNamedPipe,
		eCreateEvent,
		eCreateThread,
		eVirtualAlloc,
		eReadFile = 10,
		eWriteFile,
		eMapViewOfFile,
		eCreateFileMapping,

	eInitializeSecurityDescriptor,

	eSetSecurityDescriptorDacl,
		eCreateNamedPipe,
		eConnectNamedPipe,
		eWaitForSingleObject,
		eRegOpenKeyEx,
		eRegQueryValueEx = 20,
		ebeginthread,
		eRegEnumValue,
		eRegSetValueEx,
		eRegCreateKeyEx,
		eWaitForMultipleObjects,
		eRegisterClassEx,
		eCreateWindow,
		eCreateSemaphore,
		eFSeek,
		eFRead,
		eFWrite,
		eTmpFile,
		eSetFilePointer,
		eNew,
	};

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

	Action	m_eAction;

private:
	char m_szMsg[ERR_MSG_BUF_SIZE];
};

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

	int ErrorType() {return
ERR_TYPE_MEMORY;}
	char *ErrorText() {return
ERR_INS_MEMORY;}
};


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

```
	int ErrorType() {return
ERR_BUF_OVERFLOW;}

	char *ErrorText() {return
ERR_INS_BUF_OVERFLOW;}
};
```

ReadRegistry.cpp

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


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

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

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

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

	pReg->bCOM_SinglePool = FALSE;
	size = sizeof(szTmp);
	if ( RegQueryValueEx(hKey,
"COM_SinglePool", 0, &type, (BYTE
*)&szTmp, &size) == ERROR_SUCCESS )
	{
		if ( !stricmp(szTmp,
"YES") )
			pReg-
>bCOM_SinglePool = TRUE;
	}

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

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

		&& (type == REG_DWORD) )
		pReg-
>dwMaxPendingDeliveries = dwTmp;

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

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

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

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

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

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

	RegCloseKey(hKey);

	return FALSE;
}
```

ReadRegistry.h

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

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

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

//This structure defines the data
necessary to keep distinct for each
terminal or client connection.
typedef struct _TPCCREGISTRYDATA
{
	enum DBPROTOCOL eDB_Protocol;
	enum TXNMON eTxnMon;
	BOOL bCOM_SinglePool;
	DWORD dwMaxConnections;
	DWORD dwMaxPendingDeliveries;
	DWORD dwNumberOfDeliveryThreads;
	char szPath[128];
	char szDbServer[32];
	char szDbName[32];
	char szDbUser[32];
	char szDbPassword[32];
} TPCCREGISTRYDATA, *PTPCCREGISTRYDATA;

BOOL ReadTPCCRegistrySettings(
TPCCREGISTRYDATA *pReg );
```

Trans.h

```
/*	FILE:		TRANS.H
 *				Microsoft
TPC-C Kit Ver. 4.20.000
 *				Copyright
Microsoft, 1999
 *		All Rights Reserved
 *
 *				Version
4.10.000 audited by Richard Gimarc,
Performance Metrics, 3/17/99
 *
 *	PURPOSE:	Header file for
TPC-C structure templates.
 *
 *  Change history:
 *		4.20.000 - updated rev
number to match kit
 */
#pragma once

// String length constants
#define SERVER_NAME_LEN		20
#define DATABASE_NAME_LEN	20
#define USER_NAME_LEN		20

#define PASSWORD_LEN		20
#define TABLE_NAME_LEN		20
#define I_DATA_LEN		50
#define I_NAME_LEN		24
#define BRAND_LEN		 1
#define LAST_NAME_LEN		16
#define W_NAME_LEN		10
#define ADDRESS_LEN		20
#define STATE_LEN		 2
#define ZIP_LEN			 9
#define S_DIST_LEN		24
#define S_DATA_LEN		50
#define D_NAME_LEN		10
#define FIRST_NAME_LEN		16
#define MIDDLE_NAME_LEN		 2
#define PHONE_LEN		16
#define DATETIME_LEN		30
#define CREDIT_LEN		 2
#define C_DATA_LEN		250
#define H_DATA_LEN		24
```

Appendix A:
Source Code

```
#define DIST_INFO_LEN               24
#define MAX_OL_NEW_ORDER_ITEMS      15
#define MAX_OL_ORDER_STATUS_ITEMS   15
#define STATUS_LEN                  25
#define OL_DIST_INFO_LEN
24

// TIMESTAMP_STRUCT is provided by the
ODBC header file sqltypes.h, but is not
available
// when compiling with dblib, so
redefined here.  Note: we are using the
symbol "__SQLTYPES"
// (declared in sqltypes.h) as a way to
determine if TIMESTAMP_STRUCT has been
declared.
#ifndef __SQLTYPES
        typedef struct
        {
                        short
        /* SQLSMALLINT */       year;
                        unsigned short /*
SQLUSMALLINT */         month;
                        unsigned short /*
SQLUSMALLINT */         day;
                        unsigned short /*
SQLUSMALLINT */         hour;
                        unsigned short /*
SQLUSMALLINT */         minute;
                        unsigned short /*
SQLUSMALLINT */         second;
                        unsigned long  /*
SQLUINTEGER */ fraction;
        } TIMESTAMP_STRUCT;
#endif

// possible values for exec_status_code
after transaction completes
enum EXEC_STATUS
{
        eOK,                    // 0
        "Transaction commited."
        eInvalidItem,  // 1     "Item
number is not valid."
        eDeliveryFailed         // 2
        "Delivery Post Failed."
};

// transaction structures
typedef struct
{
        // input params
        short
        ol_supply_w_id;
        long
        ol_i_id;
        short
        ol_quantity;

        // output params
        char
        ol_i_name[I_NAME_LEN+1];
        char
        ol_brand_generic[BRAND_LEN+1];
        double
        ol_i_price;
        double
        ol_amount;
        short
        ol_stock;
} OL_NEW_ORDER_DATA;

typedef struct
{
        // input params
        short                   w_id;
        short                   d_id;
        long                    c_id;
        short                   o_ol_cnt;

        // output params
        EXEC_STATUS
        exec_status_code;
        char
        c_last[LAST_NAME_LEN+1];
        char
        c_credit[CREDIT_LEN+1];
        double                  c_discount;
        double                  w_tax;
        double                  d_tax;
        long                    o_id;
        short
        o_commit_flag;
        TIMESTAMP_STRUCT        o_entry_d;
        short
        o_all_local;
        double
        total_amount;
        OL_NEW_ORDER_DATA
        OL[MAX_OL_NEW_ORDER_ITEMS];
} NEW_ORDER_DATA, *PNEW_ORDER_DATA;

typedef struct
{       // input params
        short
w_id;
        short
d_id;
        long
c_id;
        short
c_d_id;
        short
c_w_id;
        double
h_amount;
        char
c_last[LAST_NAME_LEN+1];

        // output params

        EXEC_STATUS
exec_status_code;
        TIMESTAMP_STRUCT        h_date;
        char
w_street_1[ADDRESS_LEN+1];
        char
w_street_2[ADDRESS_LEN+1];
        char
w_city[ADDRESS_LEN+1];
        char
w_state[STATE_LEN+1];
        char
w_zip[ZIP_LEN+1];
        char
d_street_1[ADDRESS_LEN+1];
        char
d_street_2[ADDRESS_LEN+1];
        char
d_city[ADDRESS_LEN+1];
```

```
	char
d_state[STATE_LEN+1];
	char
d_zip[ZIP_LEN+1];
	char
c_first[FIRST_NAME_LEN+1];
	char
c_middle[MIDDLE_NAME_LEN + 1];
	char
c_street_1[ADDRESS_LEN+1];
	char
c_street_2[ADDRESS_LEN+1];
	char
c_city[ADDRESS_LEN+1];
	char
c_state[STATE_LEN+1];
	char
c_zip[ZIP_LEN+1];
	char
c_phone[PHONE_LEN+1];
	TIMESTAMP_STRUCT	c_since;
	char
c_credit[CREDIT_LEN+1];
	double
c_credit_lim;
	double
c_discount;
	double
c_balance;
	char
c_data[200+1];
} PAYMENT_DATA, *PPAYMENT_DATA;

typedef struct
{
	long
	ol_i_id;
	short
	ol_supply_w_id;
	short
	ol_quantity;
	double
	ol_amount;
	TIMESTAMP_STRUCT
	ol_delivery_d;
} OL_ORDER_STATUS_DATA;

typedef struct
{	// input params
	short			w_id;
	short			d_id;
	long			c_id;
	char
c_last[LAST_NAME_LEN+1];

	// output params
	EXEC_STATUS
exec_status_code;
	char
c_first[FIRST_NAME_LEN+1];
	char
c_middle[MIDDLE_NAME_LEN+1];
	double			c_balance;
	long			o_id;
	TIMESTAMP_STRUCT	o_entry_d;
	short
o_carrier_id;
	OL_ORDER_STATUS_DATA
OL[MAX_OL_ORDER_STATUS_ITEMS];
	short			o_ol_cnt;

} ORDER_STATUS_DATA, *PORDER_STATUS_DATA;


typedef struct
{	// input params
	short			w_id;
	short
	o_carrier_id;

	// output params
	EXEC_STATUS
	exec_status_code;
	SYSTEMTIME		queue_time;
	long
	o_id[10];		// id's of
delivered orders for districts 1 to 10
} DELIVERY_DATA, *PDELIVERY_DATA;

//This structure is used for posting
delivery transactions and for writing
them to the delivery server.
typedef struct _DELIVERY_TRANSACTION
{
	SYSTEMTIME	queue;
	//time delivery transaction
queued
	short		w_id;
	//delivery warehouse
	short		o_carrier_id;
	//carrier id
} DELIVERY_TRANSACTION;

typedef struct
{	// input params
	short
	w_id;
	short
	d_id;
	short
	threshold;

	// output params
	EXEC_STATUS
	exec_status_code;
	long
	low_stock;
} STOCK_LEVEL_DATA, *PSTOCK_LEVEL_DATA;
```

Txn_base.h

```
/*	FILE:		TXN_BASE.H
 *				Microsoft
TPC-C Kit Ver. 4.20.000
 *				Copyright
Microsoft, 1999
 *		All Rights Reserved
 *
 *				Version
4.10.000 audited by Richard Gimarc,
Performance Metrics, 3/17/99
 *
 *	PURPOSE:	Header file for
TPC-C txn class implementation.
 *
 *  Change history:
```

```
 *              4.20.000 - updated rev
number to match kit
 */

#pragma once

// need to declare functions for import,
unless define has already been created
// by the DLL's .cpp module for export.
#ifndef DllDecl
#define DllDecl  __declspec( dllimport )
#endif

class DllDecl CTPCC_BASE
{
        public:
                CTPCC_BASE(void) {};
                virtual ~CTPCC_BASE(void)
{};

                virtual PNEW_ORDER_DATA
                BuffAddr_NewOrder()
        = 0;
                virtual PPAYMENT_DATA
        BuffAddr_Payment()              =
0;
                virtual PDELIVERY_DATA
        BuffAddr_Delivery()             =
0;
                virtual PSTOCK_LEVEL_DATA
        BuffAddr_StockLevel()  = 0;
                virtual PORDER_STATUS_DATA
        BuffAddr_OrderStatus()= 0;

                virtual void NewOrder
        () = 0;
                virtual void Payment
        () = 0;
                virtual void Delivery
        () = 0;
                virtual void StockLevel
                () = 0;
                virtual void OrderStatus
        () = 0;
};
```

Tpcc_dblib.cpp

```
/*      FILE:           TPCC_DBLIB.CPP
 *                              Microsoft
TPC-C Kit Ver. 4.20.000
 *                              Copyright
Microsoft, 1999
 *               All Rights Reserved
 *
 *                              Version
4.10.000 audited by Richard Gimarc,
Performance Metrics, 3/17/99
 *
 *      PURPOSE:        Implements dblib
calls for TPC-C txns.
 *      Contact:        Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *              4.20.000 - updated rev
number to match kit
 *              4.10.001 - not deleting
error class in catch handler on deadlock
retry;
 *                                 not a
functional bug, but a memory leak
 *                               - had to
tweak some declarations to compile with
latest SDK; no functional change
 */

#include <windows.h>
#include <stdio.h>
#include <assert.h>

#define DBNTWIN32
#include <sqlfront.h>
#include <sqldb.h>

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

// need to declare functions for export
#define DllDecl  __declspec( dllexport )

#include "..\..\common\src\error.h"
#include "..\..\common\src\trans.h"
#include "..\..\common\src\txn_base.h"
#include "tpcc_dblib.h"

#define DEFCLPACKSIZE
        4096

// version string; must match return
value from tpcc_version stored proc
const char      sVersion[] = "4.10.000";

const                   iMaxRetries = 10;
                // how many retries on
deadlock
static  long    iConnectionCount = 0;  //
number of current dblib connections

const int iErrOleDbProvider = 7312;
const char sErrTimeoutExpired[] =
"Timeout expired";

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

        DisableThreadLibraryCalls(hModule
);
                        dbinit();
        // initialize dblib
                        break;

                case DLL_PROCESS_DETACH:
                        dbexit();
        // close all dblib
structures/connections
                        break;

                default:
                        /* nothing */;
        }
```

```
	return TRUE;
}


int err_handler(DBPROCESS *dbproc, int
severity, int dberr, int oserr, LPCSTR
dberrstr, LPCSTR oserrstr)
{
	CTPCC_DBLIB
	*pConn;

	assert(dbproc != NULL);
	pConn =
(CTPCC_DBLIB*)dbgetuserdata(dbproc);

	if (pConn != NULL)
	{
		pConn->SetDbLibError(
severity, dberr, oserr, dberrstr,
oserrstr );
	}
	return INT_CANCEL;
}

/* FUNCTION: int msg_handler(DBPROCESS
*dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext)
 *
 * PURPOSE:	This function handles DB-
Library SQL Server error messages
 *
 * ARGUMENTS:	DBPROCESS
	*dbproc			DBPROCESS
id pointer
 *				DBINT
		msgno
	message number
 *				int
			msgstate
	message state
 *				int
			severity
	message severity
 *				char
		*msgtext
	printable message description
 *
 * RETURNS:		int
		INT_CONTINUE	continue if
error is SQLETIME else INT_CANCEL action
 *
			INT_CANCEL
	cancel operation
 *
 * COMMENTS:	This function also sets
the dead lock dbproc variable if
necessary.
 *
 */

// typedef INT (SQLAPI
*DBMSGHANDLE_PROC)(PDBPROCESS, DBINT,
INT, INT, LPCSTR, LPCSTR, LPCSTR,
DBUSMALLINT);

int msg_handler(DBPROCESS *dbproc, DBINT
msgno, int msgstate, int severity,
				LPCSTR
msgtext, LPCSTR srvname, LPCSTR procname,
DBUSMALLINT line)
{
	CTPCC_DBLIB
	*pConn;

	assert(dbproc != NULL);
	pConn =
(CTPCC_DBLIB*)dbgetuserdata(dbproc);

	if (pConn != NULL)
	{
		pConn->SetSqlError( msgno,
msgstate, severity, msgtext );
	}

	return 0;
}

/* FUNCTION: void UtilStrCpy(char *
pDest, char * pSrc, int n)
 *
 * PURPOSE:	This function copies n
characters from string pSrc to pDst and
places a
 *			null character at
the end of the destination string.
 *
 * ARGUMENTS:	char
	*pDest	destination string pointer
 *				char
		*pSrc	source string
pointer
 *				int
			n
	number of characters to copy
 *
 * RETURNS:		None
 *
 * COMMENTS:	Unlike strncpy this
function ensures that the result string
is
 *				always null
terminated.
 *
 */

inline static void UtilStrCpy(char *
pDest, const BYTE * pSrc, int n)
{
    strncpy(pDest, (char *)pSrc, n);
    pDest[n] = '\0';

	return;
}

/* FUNCTION: CTPCC_DBLIB_ERR::ErrorText
 *
 */

char* CTPCC_DBLIB_ERR::ErrorText(void)
{
	int i;

	static SERRORMSG errorMsgs[] =
	{
		{ ERR_WRONG_SP_VERSION,
		"Wrong version of stored
procs on database server"	},
		{ ERR_INVALID_CUST,
		"Invalid Customer
```

```
id,name."
                        },
                { ERR_NO_SUCH_ORDER,
        "No orders found for customer."

        },
                { ERR_RETRIED_TRANS,
        "Retries before transaction
succeeded."                             },
                { 0,
                        ""


        }
        };

        static char szNotFound[] =
"Unknown error number.";

        for(i=0; errorMsgs[i].szMsg[0];
i++)
        {
                if ( m_errno ==
errorMsgs[i].iError )
                        break;
        }
        if ( !errorMsgs[i].szMsg[0] )
                return szNotFound;
        else
                return errorMsgs[i].szMsg;
}

// wrapper routine for class constructor
__declspec(dllexport) CTPCC_DBLIB*
CTPCC_DBLIB_new(
        LPCSTR szServer,                //
name of SQL server
        LPCSTR szUser,                  //
user name for login
        LPCSTR szPassword,              //
password for login
        LPCSTR szHost,                  //
workstation name; shows up in sp_who; max
30 chars, only first 10 kept by SQL
Server
        LPCSTR szDatabase )             //
name of database to use
{
        return new CTPCC_DBLIB( szServer,
szUser, szPassword, szHost, szDatabase );
}


CTPCC_DBLIB::CTPCC_DBLIB (
        LPCSTR szServer,                //
name of SQL server
        LPCSTR szUser,                  //
user name for login
        LPCSTR szPassword,              //
password for login
        LPCSTR szHost,                  //
workstation name; shows up in sp_who; max
30 chars, only first 10 kept by SQL
Server
        LPCSTR szDatabase )             //
name of database to use
{
        LOGINREC        *login;
        const BYTE      *pData;

        // initialization
        m_dbproc = NULL;
        m_DbLibErr = (CDBLIBERR*)NULL;
        m_SqlErr = (CSQLERR*)NULL;

        m_MaxRetries = 10;              //
how many retries on deadlock

        // increase max number of
connections if getting close
        if ( dbgetmaxprocs() <
(iConnectionCount+5) )
        {
                if (
dbsetmaxprocs(iConnectionCount+10) ==
FAIL )

        ThrowError(CDBLIBERR::eDbSetMaxPr
ocs);
        }

        // allocate a login structure
        login = dblogin();
        if (login == NULL)

        ThrowError(CDBLIBERR::eLogin);
        InterlockedIncrement(
&iConnectionCount );

        // register error and message
handler functions
        if (dbprocerrhandle(login,
err_handler) == NULL)

        ThrowError(CDBLIBERR::eDbProcHand
ler);

        if (dbprocmsghandle(login,
msg_handler) == NULL)

        ThrowError(CDBLIBERR::eDbProcHand
ler);


        DBSETLUSER(login, szUser);
        DBSETLPWD(login, szPassword);
        DBSETLHOST(login, szHost);
        DBSETLPACKET(login, (unsigned
short)DEFCLPACKSIZE);
        DBSETLVERSION(login, DBVER60);
                // use dblib ver 6.0
client behavior

        // set time to wait for login
        if (dbsetlogintime(60) == FAIL)

        ThrowError(CDBLIBERR::eDbSet);

        // set time to wait for statement
execution
        if (dbsettime(180) == FAIL)

        ThrowError(CDBLIBERR::eDbSet);

        m_dbproc = dbopen(login,
szServer);

        // deallocate login structure
before checking for success
        dbfreelogin( login );
```

Appendix A:
Source Code

```
	if (m_dbproc == NULL)

	ThrowError(CDBLIBERR::eDbOpen);

	// save address of class instance
so that the message and error handler
	// can get to data.
	dbsetuserdata(m_dbproc,
(LPVOID)this);

	// Use the the right database
	if (dbuse(m_dbproc, szDatabase)
== FAIL)

	ThrowError(CDBLIBERR::eDbUse);

	dbcmd(m_dbproc, "set nocount on
");			// do not return
row counts
	dbcmd(m_dbproc, "set XACT_ABORT
ON");		// rollback transaction on
abort

	if (dbsqlexec(m_dbproc) == FAIL)

	ThrowError(CDBLIBERR::eDbSqlExec)
;

	DiscardNextResults(2);

	// verify that version of stored
procs on server is correct
	dbrpcinit(m_dbproc,
"tpcc_version", 0);

	if (dbrpcexec(m_dbproc) == FAIL)

	ThrowError(CDBLIBERR::eDbRpcExec)
;

	if (dbresults(m_dbproc) !=
SUCCEED)

	ThrowError(CDBLIBERR::eDbResults)
;

	if (dbnextrow(m_dbproc) !=
REG_ROW)

	ThrowError(CDBLIBERR::eDbNextRow)
;

	char szSrvVersion[16];
	pData=dbdata(m_dbproc, 1);
	if (pData)
		UtilStrCpy(szSrvVersion,
pData, dbdatlen(m_dbproc, 1));
	else
		szSrvVersion[0]=0;
	if
(strcmp(szSrvVersion,sVersion))
		throw new CTPCC_DBLIB_ERR(
CTPCC_DBLIB_ERR::ERR_WRONG_SP_VERSION );


	DiscardNextRows(0);
	DiscardNextResults(0);
}

CTPCC_DBLIB::~CTPCC_DBLIB( void )
{
	// close db connection and
deallocate resources
	dbclose(m_dbproc);
	InterlockedDecrement(
&iConnectionCount );
	if (m_DbLibErr != NULL)
		delete m_DbLibErr;
	if (m_SqlErr != NULL)
		delete m_SqlErr;
}

void CTPCC_DBLIB::SetDbLibError(int
severity, int dberr, int oserr, LPCSTR
dberrstr, LPCSTR oserrstr)
{
	delete m_DbLibErr;
	m_DbLibErr = new
CDBLIBERR(CDBLIBERR::eUnknown, severity,
dberr, oserr);

	if (dberrstr != NULL)
	{
		m_DbLibErr->m_dberrstr =
new char[ strlen(dberrstr)+1 ];
		strcpy( m_DbLibErr-
>m_dberrstr, dberrstr );
	}

	if (oserrstr != NULL)
	{
		m_DbLibErr->m_oserrstr =
new char[ strlen(oserrstr)+1 ];
		strcpy( m_DbLibErr-
>m_oserrstr, oserrstr );
	}

}

void CTPCC_DBLIB::SetSqlError( int
/*DBINT*/ msgno, int msgstate, int
severity, LPCSTR msgtext )
{
	if (m_SqlErr == NULL)
		m_SqlErr = new CSQLERR();

	m_SqlErr->m_msgno = msgno;
	m_SqlErr->m_msgstate = msgstate;
	m_SqlErr->m_severity = severity;

	delete [] m_SqlErr->m_msgtext;
	if (msgtext != NULL)
	{
		m_SqlErr->m_msgtext = new
char[ strlen(msgtext)+1 ];
		strcpy( m_SqlErr-
>m_msgtext, msgtext );
	}
}

void CTPCC_DBLIB::ThrowError(
CDBLIBERR::ACTION eAction )
{
	// discard anything still in
return buffer
	DiscardNextRows(-1);
	DiscardNextResults(-1);
```

```
        // check for SQL Server error
first;  if yes, throw it and ignore any
DBLib error.
        if (m_SqlErr != NULL)
        {
                CSQLERR         *pSqlErr;
                pSqlErr = m_SqlErr;
                m_SqlErr = NULL;        //
clear our pointer to instance; catch
handler will delete
                throw pSqlErr;
        }

        CDBLIBERR       *pDbLibErr;
        if (m_DbLibErr == NULL)
                // this case isn't
expected to happen, since it means that
an error was returned
                // but the error handlers
were not called.
                pDbLibErr = new
CDBLIBERR(eAction);
        else
        {
                pDbLibErr = m_DbLibErr;
                pDbLibErr->m_eAction =
eAction;
                m_DbLibErr = NULL;
        // clear our pointer to instance;
catch handler will delete
        }

        throw pDbLibErr;
}

// Read and discard rows until no more.
Throw an exception if number of rows read
doesn't
// match number of rows expected.  The
row count will be ignored if the expected
count value
// passed in is negative.  A typical use
of this routine is to verify that there
are no more
// rows to be read.
void CTPCC_DBLIB::DiscardNextRows(int
iExpectedCount)
{
        int             iRowsRead = 0;
        RETCODE rc;

        while (TRUE)
        {
                rc = dbnextrow(m_dbproc);
                if (rc == NO_MORE_ROWS)
                        break;
                if (rc == FAIL)
                {
                        if (iExpectedCount
>= 0)

        ThrowError(CDBLIBERR::eDbNextRow)
;
                        else
                                break;
                }
                iRowsRead++;
        }

        if ((iExpectedCount >= 0) &&
                (iExpectedCount !=
iRowsRead))

        ThrowError(CDBLIBERR::eWrongRowCo
unt);
}

// Read and discard results until no
more.  Throw an exception if number of
result sets read doesn't
// match number expected.  The result set
count will be ignored if the expected
count value
// passed in is negative.  A typical use
of this routine is to verify that there
are no more
// result sets to be read.
void CTPCC_DBLIB::DiscardNextResults(int
iExpectedCount)
{
        int             iResultsRead = 0;
        RETCODE rc;

        while (TRUE)
        {
                rc = dbresults(m_dbproc);
                if (rc == NO_MORE_RESULTS)
                        break;
                if (rc == FAIL)
                {
                        if (iExpectedCount
>= 0)

        ThrowError(CDBLIBERR::eDbResults)
;
                        else
                                break;
                }

                DiscardNextRows(-1);
                iResultsRead++;
        }

        if ((iExpectedCount >= 0) &&
                (iExpectedCount !=
iResultsRead))

        ThrowError(CDBLIBERR::eWrongRowCo
unt);
}

void CTPCC_DBLIB::StockLevel()
{
        int                     iTryCount =
0;
        const BYTE      *pData;

        ResetError();

        while (TRUE)
        {
                try
                {

        dbrpcinit(m_dbproc,
"tpcc_stocklevel", 0);


        dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
```

```
&m_txn.StockLevel.w_id);              //
@w_id smallint

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT1, -1, -1, (BYTE *)
&m_txn.StockLevel.d_id);              //
@d_id tinyint

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
&m_txn.StockLevel.threshold);    //
@threshhold smallint

                        if
(dbrpcexec(m_dbproc) == FAIL)

        ThrowError(CDBLIBERR::eDbRpcExec)
;

                        if
(dbresults(m_dbproc) != SUCCEED)

        ThrowError(CDBLIBERR::eDbResults)
;

                        if
(dbnextrow(m_dbproc) != REG_ROW)

        ThrowError(CDBLIBERR::eDbNextRow)
;

                        if
(pData=dbdata(m_dbproc, 1))

        m_txn.StockLevel.low_stock =
*((long *) pData);


        DiscardNextRows(0);

        DiscardNextResults(0);


        m_txn.StockLevel.exec_status_code
= eOK;
                        return;
                }
                catch (CSQLERR *e)
                {
                        if ((e->m_msgno ==
1205 ||
                                (e->m_msgno
== iErrOleDbProvider &&
                                strstr(e-
>m_msgtext, sErrTimeoutExpired) != NULL))
&&

        (++iTryCount <= iMaxRetries))
                        {
                                // hit
deadlock; backoff for increasingly longer
period
                                delete e;
                                Sleep(10 *
iTryCount);
                        }
                        else
                                throw;
                }
        }       // while (TRUE)

        //if (iTryCount)
        //      throw new
CTPCC_DBLIB_ERR(CTPCC_DBLIB_ERR::ERR_RETR
IED_TRANS, iTryCount);
}


void CTPCC_DBLIB::NewOrder()
{
        int                     i;
        DBINT           commit_flag;
        DBDATETIME      datetime;
        DBDATEREC       daterec;

        int                     iTryCount =
0;
        const BYTE      *pData;

        ResetError();

        while (TRUE)
        {
                try
                {

        dbrpcinit(m_dbproc,
"tpcc_neworder", 0);


        dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
&m_txn.NewOrder.w_id);

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT1, -1, -1, (BYTE *)
&m_txn.NewOrder.d_id);

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT4, -1, -1, (BYTE *)
&m_txn.NewOrder.c_id);

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT1, -1, -1, (BYTE *)
&m_txn.NewOrder.o_ol_cnt);

                        // check whether
any order lines are for a remote
warehouse

        m_txn.NewOrder.o_all_local = 1;
                        for (i = 0; i <
m_txn.NewOrder.o_ol_cnt; i++)
                        {
                                if
(m_txn.NewOrder.OL[i].ol_supply_w_id !=
m_txn.NewOrder.w_id)
                                {

        m_txn.NewOrder.o_all_local = 0;
// at least one remote warehouse

        break;
                                }
                        }

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT1, -1, -1, (BYTE *)
&m_txn.NewOrder.o_all_local);
```

```
                    for (i = 0; i <
m_txn.NewOrder.o_ol_cnt; i++)
                    {

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT4, -1, -1, (BYTE *)
&m_txn.NewOrder.OL[i].ol_i_id);

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
&m_txn.NewOrder.OL[i].ol_supply_w_id);

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
&m_txn.NewOrder.OL[i].ol_quantity);
                    }

                    if
(dbrpcexec(m_dbproc) == FAIL)

        ThrowError(CDBLIBERR::eDbRpcExec)
;


                    // Get order line
results

        m_txn.NewOrder.total_amount = 0;
                    for (i = 0;
i<m_txn.NewOrder.o_ol_cnt; i++)
                    {
                            if
(dbresults(m_dbproc) != SUCCEED)

        ThrowError(CDBLIBERR::eDbResults)
;

                            if
(dbnumcols(m_dbproc) != 5)

        ThrowError(CDBLIBERR::eWrongNumCo
ls);

                            if
(dbnextrow(m_dbproc) != REG_ROW)

        ThrowError(CDBLIBERR::eDbNextRow)
;


        if(pData=dbdata(m_dbproc, 1))

        UtilStrCpy(m_txn.NewOrder.OL[i].o
l_i_name, pData, dbdatlen(m_dbproc, 1));

        if(pData=dbdata(m_dbproc, 2))

        m_txn.NewOrder.OL[i].ol_stock =
(*(DBSMALLINT *) pData);

        if(pData=dbdata(m_dbproc, 3))

        UtilStrCpy(m_txn.NewOrder.OL[i].o
l_brand_generic, pData,
dbdatlen(m_dbproc, 3));

        if(pData=dbdata(m_dbproc, 4))

        dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,4),

        SQLFLT8, (BYTE
*)&m_txn.NewOrder.OL[i].ol_i_price, 8);

        if(pData=dbdata(m_dbproc, 5))



        dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,5),

        SQLFLT8, (BYTE
*)&m_txn.NewOrder.OL[i].ol_amount, 8);


        m_txn.NewOrder.total_amount =
m_txn.NewOrder.total_amount +
m_txn.NewOrder.OL[i].ol_amount;


        DiscardNextRows(0);
                    }


                    // get remaining
values for w_tax, d_tax, o_id, c_last,
c_discount, c_credit, o_entry_d,
commit_flag
                    if
(dbresults(m_dbproc) != SUCCEED)

        ThrowError(CDBLIBERR::eDbResults)
;

                    if
(dbnextrow(m_dbproc) != REG_ROW)

        ThrowError(CDBLIBERR::eDbNextRow)
;

                    if
(dbnumcols(m_dbproc) != 8)

        ThrowError(CDBLIBERR::eWrongNumCo
ls);

                    if
(pData=dbdata(m_dbproc, 1))



        dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,1),
SQLFLT8, (BYTE *)&m_txn.NewOrder.w_tax,
8);
                    if
(pData=dbdata(m_dbproc, 2))



        dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,2),
SQLFLT8, (BYTE *)&m_txn.NewOrder.d_tax,
8);
                    if
(pData=dbdata(m_dbproc, 3))

        m_txn.NewOrder.o_id = (*(DBINT *)
pData);
```

```
                    if
(pData=dbdata(m_dbproc, 4))

       UtilStrCpy(m_txn.NewOrder.c_last,
pData, dbdatlen(m_dbproc, 4));
                    if
(pData=dbdata(m_dbproc, 5))

       dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,5),
SQLFLT8, (BYTE
*)&m_txn.NewOrder.c_discount, 8);
                    if
(pData=dbdata(m_dbproc, 6))

       UtilStrCpy(m_txn.NewOrder.c_credi
t, pData, dbdatlen(m_dbproc, 6));
                    if
(pData=dbdata(m_dbproc, 7))
                    {

                           datetime =
*((DBDATETIME *) pData);

       dbdatecrack(m_dbproc, &daterec,
&datetime);

       m_txn.NewOrder.o_entry_d.year   =
daterec.year;

       m_txn.NewOrder.o_entry_d.month  =
daterec.month;

       m_txn.NewOrder.o_entry_d.day    =
daterec.day;

       m_txn.NewOrder.o_entry_d.hour   =
daterec.hour;

       m_txn.NewOrder.o_entry_d.minute =
daterec.minute;

       m_txn.NewOrder.o_entry_d.second =
daterec.second;
                    }
                    if
(pData=dbdata(m_dbproc, 8))
                           commit_flag
= (*(DBTINYINT *) pData);


       DiscardNextRows(0);

       DiscardNextResults(0);


                    if (commit_flag ==
1)
                    {

       m_txn.NewOrder.total_amount *=
((1 + m_txn.NewOrder.w_tax +
m_txn.NewOrder.d_tax) * (1 -
m_txn.NewOrder.c_discount));

       m_txn.NewOrder.exec_status_code =
eOK;
                    }
                    else

       m_txn.NewOrder.exec_status_code =
eInvalidItem;

                           return;
              }
              catch (CSQLERR *e)
              {
                     if ((e->m_msgno ==
1205 ||
                            (e->m_msgno
== iErrOleDbProvider &&
                            strstr(e-
>m_msgtext, sErrTimeoutExpired) != NULL))
&&

       (++iTryCount <= iMaxRetries))
                     {
                            // hit
deadlock; backoff for increasingly longer
period
                            delete e;
                            Sleep(10 *
iTryCount);
                     }
                     else
                            throw;
              }
       }      // while (TRUE)

//     if (iTryCount)
//            throw new
CTPCC_DBLIB_ERR(CTPCC_DBLIB_ERR::ERR_RETR
IED_TRANS, iTryCount);
}


void CTPCC_DBLIB::Payment()
{
       DBDATETIME     datetime;
       DBDATEREC      daterec;

       int                   iTryCount =
0;
       const BYTE     *pData;

       ResetError();

       while (TRUE)
       {
              try
              {

       dbrpcinit(m_dbproc,
"tpcc_payment", 0);


       dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
&m_txn.Payment.w_id);

       dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
&m_txn.Payment.c_w_id);

       dbrpcparam(m_dbproc, NULL, 0,
SQLFLT8, -1, -1, (BYTE *)
&m_txn.Payment.h_amount);

       dbrpcparam(m_dbproc, NULL, 0,
```

```
SQLINT1, -1, -1, (BYTE *)
&m_txn.Payment.d_id);

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT1, -1, -1, (BYTE *)
&m_txn.Payment.c_d_id);

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT4, -1, -1, (BYTE *)
&m_txn.Payment.c_id);

                        // if customer id
is zero, then payment is by name
                        if
(m_txn.Payment.c_id == 0)

        dbrpcparam(m_dbproc, NULL, 0,
SQLCHAR, -1,
strlen(m_txn.Payment.c_last), (unsigned
char *)m_txn.Payment.c_last);


                        if
(dbrpcexec(m_dbproc) == FAIL)

        ThrowError(CDBLIBERR::eDbRpcExec)
;

                        if
(dbresults(m_dbproc) != SUCCEED)

        ThrowError(CDBLIBERR::eDbResults)
;

                        if
(dbnextrow(m_dbproc) != REG_ROW)

        ThrowError(CDBLIBERR::eDbNextRow)
;

                        if
(dbnumcols(m_dbproc) != 27)

        ThrowError(CDBLIBERR::eWrongNumCo
ls);

                        if
(pData=dbdata(m_dbproc, 1))

        m_txn.Payment.c_id = *((DBINT *)
pData);
                        if
(pData=dbdata(m_dbproc, 2))

        UtilStrCpy(m_txn.Payment.c_last,
pData, dbdatlen(m_dbproc, 2));
                        if
(pData=dbdata(m_dbproc, 3))
                        {
                                datetime =
*((DBDATETIME *) pData);

        dbdatecrack(m_dbproc, &daterec,
&datetime);

        m_txn.Payment.h_date.year   =
daterec.year;

        m_txn.Payment.h_date.month  =
daterec.month;

        m_txn.Payment.h_date.day    =
daterec.day;

        m_txn.Payment.h_date.hour   =
daterec.hour;

        m_txn.Payment.h_date.minute =
daterec.minute;

        m_txn.Payment.h_date.second =
daterec.second;
                        }
                        if
(pData=dbdata(m_dbproc, 4))

        UtilStrCpy(m_txn.Payment.w_street
_1, pData, dbdatlen(m_dbproc, 4));
                        if
(pData=dbdata(m_dbproc, 5))

        UtilStrCpy(m_txn.Payment.w_street
_2, pData, dbdatlen(m_dbproc, 5));
                        if
(pData=dbdata(m_dbproc, 6))

        UtilStrCpy(m_txn.Payment.w_city,
pData, dbdatlen(m_dbproc, 6));
                        if
(pData=dbdata(m_dbproc, 7))

        UtilStrCpy(m_txn.Payment.w_state,
pData, dbdatlen(m_dbproc, 7));
                        if
(pData=dbdata(m_dbproc, 8))

        UtilStrCpy(m_txn.Payment.w_zip,
pData, dbdatlen(m_dbproc, 8));
                        if
(pData=dbdata(m_dbproc, 9))

        UtilStrCpy(m_txn.Payment.d_street
_1, pData, dbdatlen(m_dbproc, 9));
                        if
(pData=dbdata(m_dbproc, 10))

        UtilStrCpy(m_txn.Payment.d_street
_2, pData, dbdatlen(m_dbproc, 10));
                        if
(pData=dbdata(m_dbproc, 11))

        UtilStrCpy(m_txn.Payment.d_city,
pData, dbdatlen(m_dbproc, 11));
                        if
(pData=dbdata(m_dbproc, 12))

        UtilStrCpy(m_txn.Payment.d_state,
pData, dbdatlen(m_dbproc, 12));
                        if
(pData=dbdata(m_dbproc, 13))

        UtilStrCpy(m_txn.Payment.d_zip,
pData, dbdatlen(m_dbproc, 13));
                        if
(pData=dbdata(m_dbproc, 14))

        UtilStrCpy(m_txn.Payment.c_first,
pData, dbdatlen(m_dbproc, 14));
                        if
(pData=dbdata(m_dbproc, 15))
```

```
        UtilStrCpy(m_txn.Payment.c_middle
, pData, dbdatlen(m_dbproc, 15));
                        if
(pData=dbdata(m_dbproc, 16))

        UtilStrCpy(m_txn.Payment.c_street
_1, pData, dbdatlen(m_dbproc, 16));
                        if
(pData=dbdata(m_dbproc, 17))

        UtilStrCpy(m_txn.Payment.c_street
_2, pData, dbdatlen(m_dbproc, 17));
                        if
(pData=dbdata(m_dbproc, 18))

        UtilStrCpy(m_txn.Payment.c_city,
pData, dbdatlen(m_dbproc, 18));
                        if
(pData=dbdata(m_dbproc, 19))

        UtilStrCpy(m_txn.Payment.c_state,
pData, dbdatlen(m_dbproc, 19));
                        if
(pData=dbdata(m_dbproc, 20))

        UtilStrCpy(m_txn.Payment.c_zip,
pData, dbdatlen(m_dbproc, 20));
                        if
(pData=dbdata(m_dbproc, 21))

        UtilStrCpy(m_txn.Payment.c_phone,
pData, dbdatlen(m_dbproc, 21));
                        if
(pData=dbdata(m_dbproc, 22))
                        {
                                datetime =
*((DBDATETIME *) pData);

        dbdatecrack(m_dbproc, &daterec,
&datetime);

        m_txn.Payment.c_since.year   =
daterec.year;

        m_txn.Payment.c_since.month  =
daterec.month;

        m_txn.Payment.c_since.day    =
daterec.day;

        m_txn.Payment.c_since.hour   =
daterec.hour;

        m_txn.Payment.c_since.minute =
daterec.minute;

        m_txn.Payment.c_since.second =
daterec.second;
                        }

        if(pData=dbdata(m_dbproc, 23))

        UtilStrCpy(m_txn.Payment.c_credit
, pData, dbdatlen(m_dbproc, 23));

        if(pData=dbdata(m_dbproc, 24))

        dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,24),
SQLFLT8, (BYTE
*)&m_txn.Payment.c_credit_lim, 8);

        if(pData=dbdata(m_dbproc, 25))

        dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,25),
SQLFLT8, (BYTE
*)&m_txn.Payment.c_discount, 8);

        if(pData=dbdata(m_dbproc, 26))

        dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,26),
SQLFLT8, (BYTE
*)&m_txn.Payment.c_balance, 8);

        if(pData=dbdata(m_dbproc, 27))

        UtilStrCpy(m_txn.Payment.c_data,
pData, dbdatlen(m_dbproc, 27));


        DiscardNextRows(0);

        DiscardNextResults(0);


                        if
(m_txn.Payment.c_id == 0)
                                throw new
CTPCC_DBLIB_ERR(
CTPCC_DBLIB_ERR::ERR_INVALID_CUST );
                        else

        m_txn.Payment.exec_status_code =
eOK;

                        return;
                }
                catch (CSQLERR *e)
                {
                        if ((e->m_msgno ==
1205 ||
                                (e->m_msgno
== iErrOleDbProvider &&
                                strstr(e-
>m_msgtext, sErrTimeoutExpired) != NULL))
&&

        (++iTryCount <= iMaxRetries))
                        {
                                // hit
deadlock; backoff for increasingly longer
period
                                delete e;
                                Sleep(10 *
iTryCount);
                        }
                        else
                                throw;
                }
        }       // while (TRUE)

//      if (iTryCount)
//              throw new
CTPCC_DBLIB_ERR(CTPCC_DBLIB_ERR::ERR_RETR
IED_TRANS, iTryCount);
}
```

```
void CTPCC_DBLIB::OrderStatus()
{
	int			i;
	DBDATETIME	datetime;
	DBDATEREC	daterec;

	int			iTryCount =
0;
	RETCODE		rc;
	const BYTE	*pData;

	ResetError();

	while (TRUE)
	{
		try
		{

	dbrpcinit(m_dbproc,
"tpcc_orderstatus", 0);


	dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
&m_txn.OrderStatus.w_id);

	dbrpcparam(m_dbproc, NULL, 0,
SQLINT1, -1, -1, (BYTE *)
&m_txn.OrderStatus.d_id);

	dbrpcparam(m_dbproc, NULL, 0,
SQLINT4, -1, -1, (BYTE *)
&m_txn.OrderStatus.c_id);

			// if customer id
is zero, then order status is by name
			if
(m_txn.OrderStatus.c_id == 0)

	dbrpcparam(m_dbproc, NULL, 0,
SQLCHAR, -1,
strlen(m_txn.OrderStatus.c_last),
(unsigned char
*)m_txn.OrderStatus.c_last);


			if
(dbrpcexec(m_dbproc) == FAIL)

	ThrowError(CDBLIBERR::eDbRpcExec)
;

			// Get order lines
			if
(dbresults(m_dbproc) != SUCCEED)
			{
				if
((m_DbLibErr == NULL) && (m_SqlErr ==
NULL))

	throw new CTPCC_DBLIB_ERR(
CTPCC_DBLIB_ERR::ERR_NO_SUCH_ORDER );
				else

	ThrowError(CDBLIBERR::eDbResults)
;
			}
			if
(dbnumcols(m_dbproc) != 5)

	ThrowError(CDBLIBERR::eWrongNumCo
ls);

			i = 0;
			while (TRUE)
			{
				rc =
dbnextrow(m_dbproc);
				if (rc ==
NO_MORE_ROWS)

	break;
				if (rc !=
REG_ROW)

	ThrowError(CDBLIBERR::eDbNextRow)
;


	if(pData=dbdata(m_dbproc, 1))

	m_txn.OrderStatus.OL[i].ol_supply
_w_id = (*(DBSMALLINT *) pData);

	if(pData=dbdata(m_dbproc, 2))

	m_txn.OrderStatus.OL[i].ol_i_id =
(*(DBINT *) pData);

	if(pData=dbdata(m_dbproc, 3))

	m_txn.OrderStatus.OL[i].ol_quanti
ty = (*(DBSMALLINT *) pData);

	if(pData=dbdata(m_dbproc, 4))

	dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,4),

		SQLFLT8, (BYTE
*)&m_txn.OrderStatus.OL[i].ol_amount, 8);

	if(pData=dbdata(m_dbproc, 5))
				{

	datetime = *((DBDATETIME *)
pData);

	dbdatecrack(m_dbproc, &daterec,
&datetime);

	m_txn.OrderStatus.OL[i].ol_delive
ry_d.year   = daterec.year;

	m_txn.OrderStatus.OL[i].ol_delive
ry_d.month  = daterec.month;

	m_txn.OrderStatus.OL[i].ol_delive
ry_d.day    = daterec.day;

	m_txn.OrderStatus.OL[i].ol_delive
ry_d.hour   = daterec.hour;

	m_txn.OrderStatus.OL[i].ol_delive
ry_d.minute = daterec.minute;
```

```
	m_txn.OrderStatus.OL[i].ol_delive
ry_d.second = daterec.second;
				}
				i++;
			}

	m_txn.OrderStatus.o_ol_cnt = i;


			if
(dbresults(m_dbproc) != SUCCEED)

	ThrowError(CDBLIBERR::eDbResults)
;

			if
(dbnextrow(m_dbproc) != REG_ROW)

	ThrowError(CDBLIBERR::eDbNextRow)
;

			if
(dbnumcols(m_dbproc) != 8)

	ThrowError(CDBLIBERR::eWrongNumCo
ls);


	if(pData=dbdata(m_dbproc, 1))

	m_txn.OrderStatus.c_id = (*(DBINT
*) pData);

	if(pData=dbdata(m_dbproc, 2))

	UtilStrCpy(m_txn.OrderStatus.c_la
st, pData, dbdatlen(m_dbproc,2));

	if(pData=dbdata(m_dbproc, 3))

	UtilStrCpy(m_txn.OrderStatus.c_fi
rst, pData, dbdatlen(m_dbproc,3));

	if(pData=dbdata(m_dbproc, 4))

	UtilStrCpy(m_txn.OrderStatus.c_mi
ddle, pData, dbdatlen(m_dbproc, 4));

	if(pData=dbdata(m_dbproc, 5))
			{
				datetime =
*((DBDATETIME *) pData);

	dbdatecrack(m_dbproc, &daterec,
&datetime);

	m_txn.OrderStatus.o_entry_d.year
= daterec.year;

	m_txn.OrderStatus.o_entry_d.month
= daterec.month;

	m_txn.OrderStatus.o_entry_d.day
= daterec.day;

	m_txn.OrderStatus.o_entry_d.hour
= daterec.hour;

	m_txn.OrderStatus.o_entry_d.minut
e = daterec.minute;

	m_txn.OrderStatus.o_entry_d.secon
d = daterec.second;
			}

	if(pData=dbdata(m_dbproc, 6))

	m_txn.OrderStatus.o_carrier_id =
(*(DBSMALLINT *) pData);

	if(pData=dbdata(m_dbproc, 7))

	dbconvert(m_dbproc, SQLNUMERIC,
(LPCBYTE)pData, dbdatlen(m_dbproc,7),

	  SQLFLT8, (BYTE
*)&m_txn.OrderStatus.c_balance, 8);

	if(pData=dbdata(m_dbproc, 8))

	m_txn.OrderStatus.o_id = (*(DBINT
*) pData);


	DiscardNextRows(0);

	DiscardNextResults(0);


			if
(m_txn.OrderStatus.o_ol_cnt == 0)
				throw new
CTPCC_DBLIB_ERR(
CTPCC_DBLIB_ERR::ERR_NO_SUCH_ORDER );
			else if
(m_txn.OrderStatus.c_id == 0 &&
m_txn.OrderStatus.c_last[0] == 0)
				throw new
CTPCC_DBLIB_ERR(
CTPCC_DBLIB_ERR::ERR_INVALID_CUST );
			else

	m_txn.OrderStatus.exec_status_cod
e = eOK;

			return;
		}
		catch (CSQLERR *e)
		{
			if ((e->m_msgno ==
1205 ||
				(e->m_msgno
== iErrOleDbProvider &&
				strstr(e-
>m_msgtext, sErrTimeoutExpired) != NULL))
&&

	(++iTryCount <= iMaxRetries))
			{
				// hit
deadlock; backoff for increasingly longer
period
				delete e;
				Sleep(10 *
iTryCount);
			}
			else
```

```
                              throw;
                      }
        }       // while (TRUE)

//      if (iTryCount)
//              throw new
CTPCC_DBLIB_ERR(CTPCC_DBLIB_ERR::ERR_RETR
IED_TRANS, iTryCount);
}


void CTPCC_DBLIB::Delivery()
{
        int                     i;
        int                     iTryCount =
0;
        const BYTE      *pData;


        ResetError();

        while (TRUE)
        {
                try
                {

        dbrpcinit(m_dbproc,
"tpcc_delivery", 0);


        dbrpcparam(m_dbproc, NULL, 0,
SQLINT2, -1, -1, (BYTE *)
&m_txn.Delivery.w_id);

        dbrpcparam(m_dbproc, NULL, 0,
SQLINT1, -1, -1, (BYTE *)
&m_txn.Delivery.o_carrier_id);

                        if
(dbrpcexec(m_dbproc) == FAIL)

        ThrowError(CDBLIBERR::eDbRpcExec)
;

                        if
(dbresults(m_dbproc) != SUCCEED)

        ThrowError(CDBLIBERR::eDbResults)
;

                        if
(dbnextrow(m_dbproc) != REG_ROW)

        ThrowError(CDBLIBERR::eDbNextRow)
;

                        if
(dbnumcols(m_dbproc) != 10)

        ThrowError(CDBLIBERR::eWrongNumCo
ls);

                        for (i=0; i<10;
i++)
                        {
                                if (pData =
dbdata(m_dbproc, i+1))

        m_txn.Delivery.o_id[i] = *((DBINT
*)pData);
                        }

        DiscardNextRows(0);

        DiscardNextResults(0);


        m_txn.Delivery.exec_status_code =
eOK;
                        return;
                }
                catch (CSQLERR *e)
                {
                        if ((e->m_msgno ==
1205 ||
                                (e->m_msgno
== iErrOleDbProvider &&
                                strstr(e-
>m_msgtext, sErrTimeoutExpired) != NULL))
&&

        (++iTryCount <= iMaxRetries))
                        {
                                // hit
deadlock; backoff for increasingly longer
period
                                delete e;
                                Sleep(10 *
iTryCount);
                        }
                        else
                                throw;
                }
        }       // while (TRUE)

//      if (iTryCount)
//              throw new
CTPCC_DBLIB_ERR(CTPCC_DBLIB_ERR::ERR_RETR
IED_TRANS, iTryCount);
}

void CTPCC_DBLIB::ResetError()
{
        if (m_DbLibErr != NULL)
        {
                delete m_DbLibErr;
                m_DbLibErr =
(CDBLIBERR*)NULL;
        }

        if (m_SqlErr != NULL)
        {
                delete m_SqlErr;
                m_SqlErr = (CSQLERR*)NULL;
        }
        return;
}
```

Tpcc_dblib.h

```
/*      FILE:           TPCC_DBLIB.H
 *                              Microsoft
TPC-C Kit Ver. 4.20.000
 *                              Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
```

Appendix A:
Source Code

```
 *                          Version
4.10.000 audited by Richard Gimarc,
Performance Metrics, 3/17/99
 *
 *      PURPOSE:       Header file for
TPC-C txn class implementation.
 *
 *  Change history:
 *             4.20.000 - updated rev
number to match kit
 */
#pragma once

#ifndef PDBPROCESS
#define DBPROCESS void   // dbprocess
structure type
typedef DBPROCESS * PDBPROCESS;
#endif

// need to declare functions for import,
unless define has already been created
// by the DLL's .cpp module for export.
#ifndef DllDecl
#define DllDecl  __declspec( dllimport )
#endif

class CSQLERR : public CBaseErr
{
       public:

              CSQLERR(void)
              {
                     m_msgno = 0;
                     m_msgstate = 0;
                     m_severity = 0;
                     m_msgtext = NULL;
              };

              ~CSQLERR()
              {
                     delete []
m_msgtext;
              };

              int            m_msgno;
              int            m_msgstate;
              int            m_severity;
              char   *m_msgtext;

              int ErrorType() {return
ERR_TYPE_SQL;};
              int ErrorNum() {return
m_msgno;};
              char *ErrorText() {return
m_msgtext;};

};

class CDBLIBERR : public CBaseErr
{
       public:
              enum ACTION
              {
                     eNone,
                     eUnknown,
                     eLogin,
              // error from dblogin
                     eDbOpen,
              // error from dbopen
                     eDbUse,
              // error from dbuse
                     eDbSqlExec,
              // error from dbsqlexec
                     eDbSet,
              // error from one of the
dbset* routines
                     eDbNextRow,
              // error from dbnextrow
                     eWrongRowCount,
              // more or less rows
returned than expected
                     eWrongNumCols,
       // more or less columns returned
than expected
                     eDbResults,
              // error from dbresults
                     eDbRpcExec,
              // error from dbrpcexec
                     eDbSetMaxProcs,
              // error from
dbsetmaxprocs
                     eDbProcHandler
       // error from either
dbprocerrhandle or dbprocmsghandle
              };

              CDBLIBERR(ACTION eAction,
int severity = 0, int dberror = 0, int
oserr = 0)
              {
                     m_eAction =
eAction;
                     m_severity =
severity;
                     m_dberror =
dberror;
                     m_oserr = oserr;

                     m_dberrstr = NULL;
                     m_oserrstr = NULL;
              };

              ~CDBLIBERR()
              {
                     delete []
m_dberrstr;
                     delete []
m_oserrstr;
              };

              ACTION  m_eAction;
              int            m_severity;
              int            m_dberror;
              int            m_oserr;
              char   *m_dberrstr;
              char   *m_oserrstr;

              int ErrorType() {return
ERR_TYPE_DBLIB;};
              int ErrorNum() {return
m_dberror;};
              char *ErrorText() {return
m_dberrstr;};

};

class CTPCC_DBLIB_ERR : public CBaseErr
{
       public:
```

```
		enum CTPCC_DBLIB_ERRS
		{

	ERR_WRONG_SP_VERSION = 1,	//
"Wrong version of stored procs on
database server"
			ERR_INVALID_CUST,
			// "Invalid
Customer id,name."
			ERR_NO_SUCH_ORDER,
			// "No orders
found for customer."
			ERR_RETRIED_TRANS,
			// "Retries before
transaction succeeded."
		};

		CTPCC_DBLIB_ERR( int iErr
) { m_errno = iErr; m_iTryCount = 0; };

		CTPCC_DBLIB_ERR( int iErr,
int iTryCount ) { m_errno = iErr;
m_iTryCount = iTryCount; };

		int		m_errno;
		int
	m_iTryCount;

		int ErrorType() {return
ERR_TYPE_TPCC_DBLIB;};
		int ErrorNum() {return
m_errno;};

		char *ErrorText();
};

class DllDecl CTPCC_DBLIB : public
CTPCC_BASE
{
	private:
		// declare variables and
private functions here...
		PDBPROCESS	m_dbproc;
		CDBLIBERR
	*m_DbLibErr;		// not
allocated until needed (maybe never)
		CSQLERR		*m_SqlErr;
			// not allocated
until needed (maybe never)
		int
	m_MaxRetries;		// retry
count on deadlock

		void DiscardNextRows(int
iExpectedCount);
		void
DiscardNextResults(int iExpectedCount);
		void ThrowError(
CDBLIBERR::ACTION eAction );
		void ResetError();

		union
		{
			NEW_ORDER_DATA
	NewOrder;
			PAYMENT_DATA
	Payment;
			DELIVERY_DATA
	Delivery;
			STOCK_LEVEL_DATA
	StockLevel;
			ORDER_STATUS_DATA
	OrderStatus;
		}
	m_txn;

	public:
		CTPCC_DBLIB(LPCSTR
szServer, LPCSTR szUser, LPCSTR
szPassword, LPCSTR szHost, LPCSTR
szDatabase );
		~CTPCC_DBLIB(void);

		inline PNEW_ORDER_DATA
	BuffAddr_NewOrder()		{
return &m_txn.NewOrder;	};
		inline PPAYMENT_DATA
	BuffAddr_Payment()		{
return &m_txn.Payment;	};
		inline PDELIVERY_DATA
	BuffAddr_Delivery()		{
return &m_txn.Delivery;	};
		inline PSTOCK_LEVEL_DATA
	BuffAddr_StockLevel()	{ return
&m_txn.StockLevel;	};
		inline PORDER_STATUS_DATA
	BuffAddr_OrderStatus(){ return
&m_txn.OrderStatus; };

		void NewOrder		();
		void Payment		();
		void Delivery		();
		void StockLevel
	();
		void OrderStatus	();


		// these are public
because they must be called from the
dblib err_handler and msg_hangler
		// outside of the class
		void SetDbLibError(int
severity, int dberr, int oserr, LPCSTR
dberrstr, LPCSTR oserrstr);
		void SetSqlError( int
msgno, int msgstate, int severity, LPCSTR
msgtext );

};

extern "C" DllDecl CTPCC_DBLIB*
CTPCC_DBLIB_new
	( LPCSTR szServer, LPCSTR szUser,
LPCSTR szPassword, LPCSTR szHost, LPCSTR
szDatabase );

typedef CTPCC_DBLIB*
(TYPE_CTPCC_DBLIB)(LPCSTR, LPCSTR,
LPCSTR, LPCSTR, LPCSTR);
```

Tpcc_odbc.cpp

```
/*	FILE:		TPCC_ODBC.CPP
 *				Microsoft
TPC-C Kit Ver. 4.20.000
 *				Copyright
Microsoft, 1999
```

```
 *             All Rights Reserved
 *
 *                           Version
4.10.000 audited by Richard Gimarc,
Performance Metrics, 3/17/99
 *
 *      PURPOSE:      Implements ODBC
calls for TPC-C txns.
 *      Contact:      Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *            4.20.000 - updated rev
number to match kit
 *            4.10.001 - not deleting
error class in catch handler on deadlock
retry;
 *                              not a
functional bug, but a memory leak
 */

#include <windows.h>
#include <stdio.h>
#include <assert.h>

#define DBNTWIN32
#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>

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

// need to declare functions for export
#define DllDecl  __declspec( dllexport )

#include "..\..\common\src\error.h"
#include "..\..\common\src\trans.h"
#include "..\..\common\src\txn_base.h"
#include "tpcc_odbc.h"

// version string; must match return
value from tpcc_version stored proc
const char     sVersion[] = "4.10.000";

const iMaxRetries = 10;              //
how many retries on deadlock

const int iErrOleDbProvider = 7312;
const char sErrTimeoutExpired[] =
"Timeout expired";

static SQLHENV henv = SQL_NULL_HENV;
                      // ODBC
environment handle

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

       DisableThreadLibraryCalls(hModule
);
                     if (
SQLAllocHandleStd(SQL_HANDLE_ENV,
SQL_NULL_HANDLE, &henv) != SQL_SUCCESS )
                            return
FALSE;
                     break;

              case DLL_PROCESS_DETACH:
                     if (henv != NULL)

       SQLFreeEnv(henv);
                     break;

              default:
                     /* nothing */;
       }
       return TRUE;
}

/* FUNCTION: CTPCC_ODBC_ERR::ErrorText
 *
 */

char* CTPCC_ODBC_ERR::ErrorText(void)
{
       int i;

       static SERRORMSG errorMsgs[] =
       {
              { ERR_WRONG_SP_VERSION,
              "Wrong version of stored
procs on database server"     },
              { ERR_INVALID_CUST,
              "Invalid Customer
id,name."
                     },
              { ERR_NO_SUCH_ORDER,
       "No orders found for customer."

       },
              { ERR_RETRIED_TRANS,
       "Retries before transaction
succeeded."                          },
              { 0,
                     ""


       }
       };

       static char szNotFound[] =
"Unknown error number.";

       for(i=0; errorMsgs[i].szMsg[0];
i++)
       {
              if ( m_errno ==
errorMsgs[i].iError )
                     break;
       }
       if ( !errorMsgs[i].szMsg[0] )
              return szNotFound;
       else
              return errorMsgs[i].szMsg;
}

// wrapper routine for class constructor
__declspec(dllexport) CTPCC_ODBC*
CTPCC_ODBC_new(
```

```
	LPCSTR szServer,		//
name of SQL server
	LPCSTR szUser,			//
user name for login
	LPCSTR szPassword,		//
password for login
	LPCSTR szHost,			//
not used
	LPCSTR szDatabase )		//
name of database to use
{
	return new CTPCC_ODBC( szServer,
szUser, szPassword, szHost, szDatabase );
}


CTPCC_ODBC::CTPCC_ODBC (
	LPCSTR szServer,
		// name of SQL server
	LPCSTR szUser,
		// user name for login
	LPCSTR szPassword,
		// password for login
	LPCSTR szHost,
		// not used
	LPCSTR szDatabase
		// name of database to use
	)
{
	RETCODE		rc;

	// initialization
	m_hdbc = SQL_NULL_HDBC;
	m_hstmt = SQL_NULL_HSTMT;

	m_hstmtNewOrder = SQL_NULL_HSTMT;
	m_hstmtPayment = SQL_NULL_HSTMT;
	m_hstmtDelivery = SQL_NULL_HSTMT;
	m_hstmtOrderStatus =
SQL_NULL_HSTMT;
	m_hstmtStockLevel =
SQL_NULL_HSTMT;

	m_descNewOrderCols1 =
SQL_NULL_HDESC;
	m_descNewOrderCols2 =
SQL_NULL_HDESC;
	m_descOrderStatusCols1 =
SQL_NULL_HDESC;
	m_descOrderStatusCols2 =
SQL_NULL_HDESC;


	if (
SQLAllocHandle(SQL_HANDLE_DBC, henv,
&m_hdbc) != SQL_SUCCESS )

	ThrowError(CODBCERR::eAllocHandle
);

	if ( SQLSetConnectOption(m_hdbc,
SQL_PACKET_SIZE, 4096) != SQL_SUCCESS )

	ThrowError(CODBCERR::eConnOption)
;

	{
		char
	szConnectStr[256];
		char
	szOutStr[1024];
		SQLSMALLINT
	iOutStrLen;

		sprintf( szConnectStr,
"DRIVER=SQL
Server;SERVER=%s;UID=%s;PWD=%s;DATABASE=%
s",
			szServer, szUser,
szPassword, szDatabase );

		rc =
SQLDriverConnect(m_hdbc, NULL,
(SQLCHAR*)szConnectStr,
sizeof(szConnectStr),

	(SQLCHAR*)szOutStr,
sizeof(szOutStr), &iOutStrLen,
SQL_DRIVER_NOPROMPT );

		if (rc != SQL_SUCCESS &&
rc != SQL_SUCCESS_WITH_INFO)

	ThrowError(CODBCERR::eConnect);
	}

	if
(SQLAllocHandle(SQL_HANDLE_STMT, m_hdbc,
&m_hstmt) != SQL_SUCCESS)

	ThrowError(CODBCERR::eAllocHandle
);

	{
		char
	buffer[128];

		// set some options
affecting connection behavior
		strcpy(buffer, "set
nocount on set XACT_ABORT ON");
		rc =
SQLExecDirect(m_hstmt, (unsigned char
*)buffer, SQL_NTS);
		if (rc != SQL_SUCCESS &&
rc != SQL_SUCCESS_WITH_INFO)

	ThrowError(CODBCERR::eExecDirect)
;

		// verify that version of
stored procs on server is correct
		char db_sp_version[10];
		strcpy(buffer, "{call
tpcc_version}");
		rc =
SQLExecDirect(m_hstmt, (unsigned char
*)buffer, SQL_NTS);
		if (rc != SQL_SUCCESS &&
rc != SQL_SUCCESS_WITH_INFO)

	ThrowError(CODBCERR::eExecDirect)
;
		if ( SQLBindCol(m_hstmt,
1, SQL_C_CHAR, &db_sp_version,
sizeof(db_sp_version), NULL) !=
SQL_SUCCESS )
```

```
	int i = 0;
	if (	SQLBindParameter(m_hstmt,
++i, SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0,
&m_txn.StockLevel.w_id, 0, NULL) !=
SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT,
SQL_TINYINT, 0, 0,
&m_txn.StockLevel.d_id, 0, NULL) !=
SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0,
&m_txn.StockLevel.threshold, 0, NULL) !=
SQL_SUCCESS
		)

	ThrowError(CODBCERR::eBindParam);

	if ( SQLBindCol(m_hstmt, 1,
SQL_C_SLONG, &m_txn.StockLevel.low_stock,
0, NULL) != SQL_SUCCESS )

	ThrowError(CODBCERR::eBindCol);
}

void CTPCC_ODBC::StockLevel()
{
	RETCODE		rc;
	int			iTryCount =
0;

	m_hstmt = m_hstmtStockLevel;

	while (TRUE)
	{
		try
		{
			rc =
SQLExecDirectW(m_hstmt,
(SQLWCHAR*)L"{call
tpcc_stocklevel(?,?,?)}", SQL_NTS);
			if (rc !=
SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)

	ThrowError(CODBCERR::eExecDirect)
;

			if (
SQLFetch(m_hstmt) == SQL_ERROR )

	ThrowError(CODBCERR::eFetch);


	SQLFreeStmt(m_hstmt, SQL_CLOSE);


	m_txn.StockLevel.exec_status_code
= eOK;
			break;
		}
		catch (CODBCERR *e)
		{
			if ((!e-
>m_bDeadLock) || (++iTryCount >
iMaxRetries))
					throw;

				// hit deadlock;
backoff for increasingly longer period
				delete e;
				Sleep(10 *
iTryCount);
			}
	}

//	if (iTryCount)
//		throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIE
D_TRANS, iTryCount);
}

void CTPCC_ODBC::InitNewOrderParams()
{
	if (
SQLAllocHandle(SQL_HANDLE_STMT, m_hdbc,
&m_hstmtNewOrder) != SQL_SUCCESS
		||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descNewOrderCols1) != SQL_SUCCESS
		||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descNewOrderCols2) != SQL_SUCCESS
		)

	ThrowError(CODBCERR::eAllocHandle
);

	m_hstmt = m_hstmtNewOrder;

	if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descNewOrderCols1, SQL_IS_POINTER ) !=
SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);

	int i = 0;
	if (	SQLBindParameter(m_hstmt,
++i, SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0, &m_txn.NewOrder.w_id,
0, NULL) != SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT,
SQL_TINYINT, 0, 0, &m_txn.NewOrder.d_id,
0, NULL) != SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG,
SQL_INTEGER, 0, 0, &m_txn.NewOrder.c_id,
0, NULL) != SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT,
SQL_TINYINT, 0, 0,
&m_txn.NewOrder.o_ol_cnt, 0, NULL) !=
SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT,
SQL_TINYINT, 0, 0,
&m_txn.NewOrder.o_all_local, 0, NULL) !=
SQL_SUCCESS
		)
```

```
	ThrowError(CODBCERR::eBindParam);

	for (int j=0;
j<MAX_OL_NEW_ORDER_ITEMS; j++)
	{
		if (
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG,
SQL_INTEGER, 0, 0,
&m_txn.NewOrder.OL[j].ol_i_id, 0, NULL)
!= SQL_SUCCESS
			||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0,
&m_txn.NewOrder.OL[j].ol_supply_w_id, 0,
NULL) != SQL_SUCCESS
			||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0,
&m_txn.NewOrder.OL[j].ol_quantity, 0,
NULL) != SQL_SUCCESS
			)

	ThrowError(CODBCERR::eBindParam);
	}

	// set the bind offset pointer
	if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_ROW_BIND_OFFSET_PTR,
&m_BindOffset, SQL_IS_POINTER ) !=
SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);

	i = 0;
	if (   SQLBindCol(m_hstmt, ++i,
SQL_C_CHAR,
&m_txn.NewOrder.OL[0].ol_i_name,
sizeof(m_txn.NewOrder.OL[0].ol_i_name),
NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_SSHORT,
&m_txn.NewOrder.OL[0].ol_stock, 0, NULL)
!= SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.NewOrder.OL[0].ol_brand_generic,
sizeof(m_txn.NewOrder.OL[0].ol_brand_gene
ric), NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.NewOrder.OL[0].ol_i_price, 0,
NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.NewOrder.OL[0].ol_amount, 0, NULL)
!= SQL_SUCCESS
		)

	ThrowError(CODBCERR::eBindCol);

	// associate the column bindings
for the second result set
	if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descNewOrderCols2, SQL_IS_POINTER ) !=
SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);

	i = 0;
	if (   SQLBindCol(m_hstmt, ++i,
SQL_C_DOUBLE,    &m_txn.NewOrder.w_tax,
0, NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.NewOrder.d_tax, 0, NULL) !=
SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_SLONG,
&m_txn.NewOrder.o_id, 0, NULL) !=
SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.NewOrder.c_last,
sizeof(m_txn.NewOrder.c_last), NULL) !=
SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.NewOrder.c_discount, 0, NULL) !=
SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.NewOrder.c_credit,
sizeof(m_txn.NewOrder.c_credit), NULL) !=
SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_TYPE_TIMESTAMP,
&m_txn.NewOrder.o_entry_d, 0, NULL) !=
SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_SLONG,     &m_no_commit_flag,
0, NULL) != SQL_SUCCESS
		)

	ThrowError(CODBCERR::eBindCol);
}

void CTPCC_ODBC::NewOrder()
{
	int
	i;
	RETCODE				rc;
	int
	iTryCount = 0;

						//
0         1         2

						//
012345678901234567890123456789
	wchar_t
	szSqlTemplate[] = L"{call
tpcc_neworder(?,?,?,?,?,"

L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,"

L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,"

L"?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)}";
```

```
	m_hstmt = m_hstmtNewOrder;

	// associate the parameter and
column bindings for this transaction
	if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descNewOrderCols1, SQL_IS_POINTER ) !=
SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);

	// clip statement buffer based on
number of parameters
	// fixed part is 29 chars and
variable part is 6 chars per line item
	i = 29 +
m_txn.NewOrder.o_ol_cnt*6;
	wcscpy( &szSqlTemplate[i], L")}"
);

	// check whether any order lines
are for a remote warehouse
	m_txn.NewOrder.o_all_local = 1;

	for (i = 0; i <
m_txn.NewOrder.o_ol_cnt; i++)
	{
		if
(m_txn.NewOrder.OL[i].ol_supply_w_id !=
m_txn.NewOrder.w_id)
		{

	m_txn.NewOrder.o_all_local = 0;
// at least one remote warehouse
			break;
		}
	}

	while (TRUE)
	{
		try
		{
			m_BindOffset = 0;
			rc =
SQLExecDirectW(m_hstmt,
(SQLWCHAR*)szSqlTemplate, SQL_NTS);
			if (rc !=
SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)

	ThrowError(CODBCERR::eExecDirect)
;

			// Get order line
results

	m_txn.NewOrder.total_amount = 0;
			for (i = 0;
i<m_txn.NewOrder.o_ol_cnt; i++)
			{
				// set the
bind offset value...

	m_BindOffset = i *
sizeof(m_txn.NewOrder.OL[0]);

				if (
SQLFetch(m_hstmt) == SQL_ERROR)

	ThrowError(CODBCERR::eFetch);

					// move to
the next resultset
					if (
SQLMoreResults(m_hstmt) == SQL_ERROR )

	ThrowError(CODBCERR::eMoreResults
);


	m_txn.NewOrder.total_amount +=
m_txn.NewOrder.OL[i].ol_amount;
			}

			// associate the
column bindings for the second result set
			if (
SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descNewOrderCols2, SQL_IS_POINTER ) !=
SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);


			if (
SQLFetch(m_hstmt) == SQL_ERROR)

	ThrowError(CODBCERR::eFetch);


	SQLFreeStmt(m_hstmt, SQL_CLOSE);

			if
(m_no_commit_flag == 1)
			{

	m_txn.NewOrder.total_amount *=
((1 + m_txn.NewOrder.w_tax +
m_txn.NewOrder.d_tax) * (1 -
m_txn.NewOrder.c_discount));

	m_txn.NewOrder.exec_status_code =
eOK;
			}
			else

	m_txn.NewOrder.exec_status_code =
eInvalidItem;

			break;
		}
		catch (CODBCERR *e)
		{
			if ((!e-
>m_bDeadLock) || (++iTryCount >
iMaxRetries))
				throw;

			// hit deadlock;
backoff for increasingly longer period
			delete e;
			Sleep(10 *
iTryCount);
		}
	}

//	if (iTryCount)
```

```
//              throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIE
D_TRANS, iTryCount);
}

void CTPCC_ODBC::InitPaymentParams()
{
        if (
SQLAllocHandle(SQL_HANDLE_STMT, m_hdbc,
&m_hstmtPayment) != SQL_SUCCESS )

        ThrowError(CODBCERR::eAllocHandle
);

        m_hstmt = m_hstmtPayment;

        int i = 0;
        if (    SQLBindParameter(m_hstmt,
++i, SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0, &m_txn.Payment.w_id,
0, NULL) != SQL_SUCCESS
                ||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0,
&m_txn.Payment.c_w_id, 0, NULL) !=
SQL_SUCCESS
                ||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_DOUBLE,
SQL_NUMERIC, 6, 2,
&m_txn.Payment.h_amount, 0, NULL) !=
SQL_SUCCESS
                ||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT,
SQL_TINYINT, 0, 0, &m_txn.Payment.d_id,
0, NULL) != SQL_SUCCESS
                ||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT,
SQL_TINYINT, 0, 0, &m_txn.Payment.c_d_id,
0, NULL) != SQL_SUCCESS
                ||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG,
SQL_INTEGER, 0, 0, &m_txn.Payment.c_id,
0, NULL) != SQL_SUCCESS
                ||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR,
sizeof(m_txn.Payment.c_last), 0,
&m_txn.Payment.c_last,
sizeof(m_txn.Payment.c_last), NULL) !=
SQL_SUCCESS
                )

        ThrowError(CODBCERR::eBindParam);

        i = 0;
        if (    SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG,     &m_txn.Payment.c_id,
        0, NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_last,
        sizeof(m_txn.Payment.c_last),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_TYPE_TIMESTAMP,
&m_txn.Payment.h_date,        0, NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.w_street_1,
        sizeof(m_txn.Payment.w_street_1),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.w_street_2,
        sizeof(m_txn.Payment.w_street_2),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.w_city,
        sizeof(m_txn.Payment.w_city),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.w_state,
        sizeof(m_txn.Payment.w_state),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.w_zip,
        sizeof(m_txn.Payment.w_zip),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.d_street_1,
        sizeof(m_txn.Payment.d_street_1),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.d_street_2,
        sizeof(m_txn.Payment.d_street_2),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.d_city,
        sizeof(m_txn.Payment.d_city),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.d_state,
        sizeof(m_txn.Payment.d_state),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.d_zip,
        sizeof(m_txn.Payment.d_zip),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_first,
        sizeof(m_txn.Payment.c_first),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_middle,
        sizeof(m_txn.Payment.c_middle),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_street_1,
        sizeof(m_txn.Payment.c_street_1),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_street_2,
```

```
        sizeof(m_txn.Payment.c_street_2),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_city,
        sizeof(m_txn.Payment.c_city),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_state,
        sizeof(m_txn.Payment.c_state),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_zip,
        sizeof(m_txn.Payment.c_zip),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_phone,
        sizeof(m_txn.Payment.c_phone),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_TYPE_TIMESTAMP,
&m_txn.Payment.c_since,              0,
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_credit,
        sizeof(m_txn.Payment.c_credit),
NULL) != SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.Payment.c_credit_lim,0, NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.Payment.c_discount,    0, NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.Payment.c_balance,     0, NULL) !=
SQL_SUCCESS
                || SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.Payment.c_data,
        sizeof(m_txn.Payment.c_data),
NULL) != SQL_SUCCESS
                )

        ThrowError(CODBCERR::eBindCol);
}

void CTPCC_ODBC::Payment()
{
        RETCODE         rc;
        int                     iTryCount =
0;

        m_hstmt = m_hstmtPayment;

        if (m_txn.Payment.c_id != 0)
                m_txn.Payment.c_last[0] =
0;

        while (TRUE)
        {
                try
                {
                        rc =
SQLExecDirectW(m_hstmt,
(SQLWCHAR*)L"{call
tpcc_payment(?,?,?,?,?,?,?,?)}", SQL_NTS);
                        if (rc !=
SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)

        ThrowError(CODBCERR::eExecDirect)
;

                        if (
SQLFetch(m_hstmt) == SQL_ERROR)

        ThrowError(CODBCERR::eFetch);


        SQLFreeStmt(m_hstmt, SQL_CLOSE);

                        if
(m_txn.Payment.c_id == 0)
                                throw new
CTPCC_ODBC_ERR(
CTPCC_ODBC_ERR::ERR_INVALID_CUST );
                        else

        m_txn.Payment.exec_status_code =
eOK;

                        break;
                }
                catch (CODBCERR *e)
                {
                        if ((!e-
>m_bDeadLock) || (++iTryCount >
iMaxRetries))
                                throw;

                        // hit deadlock;
backoff for increasingly longer period
                        delete e;
                        Sleep(10 *
iTryCount);
                }
        }

//      if (iTryCount)
//              throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIE
D_TRANS, iTryCount);
}

void CTPCC_ODBC::InitOrderStatusParams()
{
        if (
SQLAllocHandle(SQL_HANDLE_STMT, m_hdbc,
&m_hstmtOrderStatus) != SQL_SUCCESS
                ||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descOrderStatusCols1) != SQL_SUCCESS
                ||
SQLAllocHandle(SQL_HANDLE_DESC, m_hdbc,
&m_descOrderStatusCols2) != SQL_SUCCESS
                )

        ThrowError(CODBCERR::eAllocHandle
);

        m_hstmt = m_hstmtOrderStatus;
```

```
	if (	SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descOrderStatusCols1, SQL_IS_POINTER )
!= SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);


	int i = 0;
	if (	SQLBindParameter(m_hstmt,
++i, SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0,
&m_txn.OrderStatus.w_id, 0, NULL) !=
SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_UTINYINT,
SQL_TINYINT, 0, 0,
&m_txn.OrderStatus.d_id, 0, NULL) !=
SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SLONG,
SQL_INTEGER, 0, 0,
&m_txn.OrderStatus.c_id, 0, NULL) !=
SQL_SUCCESS
		||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_CHAR, SQL_CHAR,
sizeof(m_txn.OrderStatus.c_last), 0,
&m_txn.OrderStatus.c_last,
sizeof(m_txn.OrderStatus.c_last), NULL)
!= SQL_SUCCESS
		)

	ThrowError(CODBCERR::eBindParam);

	// configure block cursor
	if (	SQLSetStmtAttrW(m_hstmt,
SQL_ATTR_ROW_BIND_TYPE,
(SQLPOINTER)sizeof(m_txn.OrderStatus.OL[0
]), 0) != SQL_SUCCESS
		||
SQLSetStmtAttrW(m_hstmt,
SQL_ATTR_ROWS_FETCHED_PTR,
&m_RowsFetched, 0) != SQL_SUCCESS
		)

	ThrowError(CODBCERR::eSetStmtAttr
);

	i = 0;
	if (	SQLBindCol(m_hstmt, ++i,
SQL_C_SSHORT,
&m_txn.OrderStatus.OL[0].ol_supply_w_id,
0, NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_SLONG,
&m_txn.OrderStatus.OL[0].ol_i_id, 0,
NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_SSHORT,
&m_txn.OrderStatus.OL[0].ol_quantity, 0,
NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.OrderStatus.OL[0].ol_amount, 0,
NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_TYPE_TIMESTAMP,
&m_txn.OrderStatus.OL[0].ol_delivery_d,
0, NULL) != SQL_SUCCESS
		)

	ThrowError(CODBCERR::eBindCol);


	if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descOrderStatusCols2, SQL_IS_POINTER )
!= SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);

	i = 0;
	if (	SQLBindCol(m_hstmt, ++i,
SQL_C_SLONG, &m_txn.OrderStatus.c_id, 0,
NULL) != SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.OrderStatus.c_last,
sizeof(m_txn.OrderStatus.c_last), NULL)
!= SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.OrderStatus.c_first,
sizeof(m_txn.OrderStatus.c_first), NULL)
!= SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_CHAR,
&m_txn.OrderStatus.c_middle,
sizeof(m_txn.OrderStatus.c_middle), NULL)
!= SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_TYPE_TIMESTAMP,
&m_txn.OrderStatus.o_entry_d, 0, NULL) !=
SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_SSHORT,
&m_txn.OrderStatus.o_carrier_id, 0, NULL)
!= SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_DOUBLE,
&m_txn.OrderStatus.c_balance, 0, NULL) !=
SQL_SUCCESS
		|| SQLBindCol(m_hstmt,
++i, SQL_C_SLONG,
&m_txn.OrderStatus.o_id, 0, NULL) !=
SQL_SUCCESS
		)

	ThrowError(CODBCERR::eBindCol);
}

void CTPCC_ODBC::OrderStatus()
{
	int
		iTryCount = 0;
	RETCODE
	rc;

	m_hstmt = m_hstmtOrderStatus;

	if ( SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descOrderStatusCols1, SQL_IS_POINTER )
!= SQL_SUCCESS )
```

```
	ThrowError(CODBCERR::eSetStmtAttr
);

	if (m_txn.OrderStatus.c_id != 0)

	m_txn.OrderStatus.c_last[0] = 0;

	while (TRUE)
	{
		try
		{
			// configure block
cursor
			if (
SQLSetStmtAttrW(m_hstmt,
SQL_ATTR_ROW_ARRAY_SIZE, (SQLPOINTER)1,
0) != SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);

			rc =
SQLExecDirectW(m_hstmt,
(SQLWCHAR*)L"{call
tpcc_orderstatus(?,?,?,?)}", SQL_NTS);
			if ( ((rc ==
SQL_SUCCESS_WITH_INFO) && (m_RowsFetched
!= 0)) || (rc == SQL_ERROR) )

	ThrowError(CODBCERR::eExecDirect)
;

			// configure block
cursor
			if (
SQLSetStmtAttrW(m_hstmt,
SQL_ATTR_ROW_ARRAY_SIZE,
(SQLPOINTER)MAX_OL_ORDER_STATUS_ITEMS, 0)
!= SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);

			rc =
SQLFetchScroll( m_hstmt, SQL_FETCH_NEXT,
0 );
			if ( ((rc ==
SQL_SUCCESS_WITH_INFO) && (m_RowsFetched
!= 0)) || (rc == SQL_ERROR) )

	ThrowError(CODBCERR::eFetchScroll
);


	m_txn.OrderStatus.o_ol_cnt =
(short)m_RowsFetched;

			if
(m_txn.OrderStatus.o_ol_cnt != 0)
			{
				if (
SQLSetStmtAttrW( m_hstmt,
SQL_ATTR_APP_ROW_DESC,
m_descOrderStatusCols2, SQL_IS_POINTER )
!= SQL_SUCCESS )

	ThrowError(CODBCERR::eSetStmtAttr
);
				if (
SQLMoreResults(m_hstmt) == SQL_ERROR )

	ThrowError(CODBCERR::eMoreResults
);

				if ( (rc =
SQLFetch(m_hstmt)) == SQL_ERROR)

	ThrowError(CODBCERR::eFetch);
			}


	SQLFreeStmt(m_hstmt, SQL_CLOSE);

			if
(m_txn.OrderStatus.o_ol_cnt == 0)
				throw new
CTPCC_ODBC_ERR(
CTPCC_ODBC_ERR::ERR_NO_SUCH_ORDER );
			else if
(m_txn.OrderStatus.c_id == 0 &&
m_txn.OrderStatus.c_last[0] == 0)
				throw new
CTPCC_ODBC_ERR(
CTPCC_ODBC_ERR::ERR_INVALID_CUST );
			else

	m_txn.OrderStatus.exec_status_cod
e = eOK;

			break;
		}
		catch (CODBCERR *e)
		{
			if ((!e-
>m_bDeadLock) || (++iTryCount >
iMaxRetries))
				throw;

			// hit deadlock;
backoff for increasingly longer period
			delete e;
			Sleep(10 *
iTryCount);
		}
	}

//	if (iTryCount)
//		throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIE
D_TRANS, iTryCount);
}

void CTPCC_ODBC::InitDeliveryParams()
{
	if (
SQLAllocHandle(SQL_HANDLE_STMT, m_hdbc,
&m_hstmtDelivery) != SQL_SUCCESS )

	ThrowError(CODBCERR::eAllocHandle
);

	m_hstmt = m_hstmtDelivery;


	int i = 0;
	if (   SQLBindParameter(m_hstmt,
++i, SQL_PARAM_INPUT, SQL_C_SSHORT,
```

```
SQL_SMALLINT, 0, 0, &m_txn.Delivery.w_id,
0, NULL) != SQL_SUCCESS
                ||
SQLBindParameter(m_hstmt, ++i,
SQL_PARAM_INPUT, SQL_C_SSHORT,
SQL_SMALLINT, 0, 0,
&m_txn.Delivery.o_carrier_id, 0, NULL) !=
SQL_SUCCESS
                )

        ThrowError(CODBCERR::eBindParam);

        for (i=0;i<10;i++)
        {
                if ( SQLBindCol(m_hstmt,
(UWORD)(i+1), SQL_C_SLONG,
&m_txn.Delivery.o_id[i], 0, NULL) !=
SQL_SUCCESS )

        ThrowError(CODBCERR::eBindCol);
        }
}

void CTPCC_ODBC::Delivery()
{
        RETCODE         rc;
        int                     iTryCount =
0;

        m_hstmt = m_hstmtDelivery;

        while (TRUE)
        {
                try
                {
                        rc =
SQLExecDirectW(m_hstmt,
(SQLWCHAR*)L"{call tpcc_delivery(?,?)}",
SQL_NTS);
                        if (rc !=
SQL_SUCCESS && rc !=
SQL_SUCCESS_WITH_INFO)

        ThrowError(CODBCERR::eExecDirect)
;

                        if (
SQLFetch(m_hstmt) == SQL_ERROR )

        ThrowError(CODBCERR::eFetch);


        SQLFreeStmt(m_hstmt, SQL_CLOSE);

        m_txn.Delivery.exec_status_code =
eOK;
                        break;
                }
                catch (CODBCERR *e)
                {
                        if ((!e-
>m_bDeadLock) || (++iTryCount >
iMaxRetries))
                                throw;

                        // hit deadlock;
backoff for increasingly longer period
                        delete e;
                        Sleep(10 *
iTryCount);
                }
        }

//      if (iTryCount)
//              throw new
CTPCC_ODBC_ERR(CTPCC_ODBC_ERR::ERR_RETRIE
D_TRANS, iTryCount);
}
```

Tpcc_odbc.h

```
/*      FILE:           TPCC_ODBC.H
 *                              Microsoft
TPC-C Kit Ver. 4.20.000
 *                              Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                              Version
4.10.000 audited by Richard Gimarc,
Performance Metrics, 3/17/99
 *
 *      PURPOSE:        Header file for
TPC-C txn class implementation.
 *
 *  Change history:
 *              4.20.000 - updated rev
number to match kit
 */
#pragma once

// need to declare functions for import,
unless define has already been created
// by the DLL's .cpp module for export.
#ifndef DllDecl
#define DllDecl  __declspec( dllimport )
#endif

class CODBCERR : public CBaseErr
{
        public:
                enum ACTION
                {
                        eNone,
                        eUnknown,
                        eAllocConn,
                // error from
SQLAllocConnect
                        eAllocHandle,
        // error from SQLAllocHandle
                        eConnOption,
        // error from SQLSetConnectOption
                        eConnect,
                // error from SQLConnect
                        eAllocStmt,
                // error from SQLAllocStmt
                        eExecDirect,
        // error from SQLExecDirect
                        eBindParam,
                // error from
SQLBindParameter
                        eBindCol,
                // error from SQLBindCol
                        eFetch,
                // error from SQLFetch
                        eFetchScroll,
        // error from SQLFetchScroll
```

```
			eMoreResults,
	// error from SQLMoreResults
			ePrepare,
		// error from SQLPrepare
			eExecute,
		// error from SQLExecute
			eSetEnvAttr,
	// error from SQLSetEnvAttr
			eSetStmtAttr
	// error from SQLSetStmtAttr
		};

		CODBCERR(void)
		{
			m_eAction = eNone;
			m_NativeError = 0;
			m_bDeadLock =
FALSE;
			m_odbcerrstr =
NULL;
		};

		~CODBCERR()
		{
			if (m_odbcerrstr
!= NULL)
				delete []
m_odbcerrstr;
		};

		ACTION	m_eAction;
		int
	m_NativeError;
		BOOL	m_bDeadLock;
		char	*m_odbcerrstr;

		int ErrorType() {return
ERR_TYPE_ODBC;};
		int ErrorNum() {return
m_NativeError;};
		char *ErrorText() {return
m_odbcerrstr;};

};


class CTPCC_ODBC_ERR : public CBaseErr
{
	public:
		enum TPCC_ODBC_ERRS
		{

	ERR_WRONG_SP_VERSION = 1,	//
"Wrong version of stored procs on
database server"
			ERR_INVALID_CUST,
			// "Invalid
Customer id,name."
			ERR_NO_SUCH_ORDER,
			// "No orders
found for customer."
			ERR_RETRIED_TRANS,
			// "Retries before
transaction succeeded."
		};

		CTPCC_ODBC_ERR( int iErr )
{ m_errno = iErr; m_iTryCount = 0; };
			CTPCC_ODBC_ERR( int iErr,
int iTryCount ) { m_errno = iErr;
m_iTryCount = iTryCount; };

			int		m_errno;
			int
		m_iTryCount;

			int ErrorType() {return
ERR_TYPE_TPCC_ODBC;};
			int ErrorNum() {return
m_errno;};

			char *ErrorText();
};

class DllDecl CTPCC_ODBC : public
CTPCC_BASE
{
		private:
			// declare variables and
private functions here...
			BOOL
		m_bDeadlock;		//
transaction was selected as deadlock
victim
			int
		m_MaxRetries;		// retry
count on deadlock

			SQLHENV		m_henv;
				// ODBC
environment handle
			SQLHDBC		m_hdbc;
			SQLHSTMT	m_hstmt;
				// the current
hstmt

			SQLHSTMT
		m_hstmtNewOrder;
			SQLHSTMT
		m_hstmtPayment;
			SQLHSTMT
		m_hstmtDelivery;
			SQLHSTMT
		m_hstmtOrderStatus;
			SQLHSTMT
		m_hstmtStockLevel;

			SQLHDESC
		m_descNewOrderCols1;
			SQLHDESC
		m_descNewOrderCols2;
			SQLHDESC
		m_descOrderStatusCols1;
			SQLHDESC
		m_descOrderStatusCols2;

			// new-order specific
fields
			SQLUINTEGER
		m_BindOffset;
			SQLUINTEGER
		m_RowsFetched;
			int
		m_no_commit_flag;

			void ThrowError(
CODBCERR::ACTION eAction );
```

```
		void InitNewOrderParams();
		void InitPaymentParams();
		void InitDeliveryParams();
		void
InitStockLevelParams();
		void
InitOrderStatusParams();

		union
		{
			NEW_ORDER_DATA
	NewOrder;
			PAYMENT_DATA
	Payment;
			DELIVERY_DATA
	Delivery;
			STOCK_LEVEL_DATA
	StockLevel;
			ORDER_STATUS_DATA
	OrderStatus;
		}
	m_txn;

	public:
		CTPCC_ODBC(LPCSTR
szServer, LPCSTR szUser, LPCSTR
szPassword, LPCSTR szHost, LPCSTR
szDatabase);
		~CTPCC_ODBC(void);

		inline PNEW_ORDER_DATA
	BuffAddr_NewOrder()		{
return &m_txn.NewOrder;	};
		inline PPAYMENT_DATA
	BuffAddr_Payment()		{
return &m_txn.Payment;	};
		inline PDELIVERY_DATA
	BuffAddr_Delivery()		{
return &m_txn.Delivery;	};
		inline PSTOCK_LEVEL_DATA
	BuffAddr_StockLevel()	{ return
&m_txn.StockLevel;	};
		inline PORDER_STATUS_DATA
	BuffAddr_OrderStatus()	{ return
&m_txn.OrderStatus; };

		void NewOrder		();
		void Payment		();
		void Delivery		();
		void StockLevel
	();
		void OrderStatus	();

};


// wrapper routine for class constructor
extern "C" DllDecl CTPCC_ODBC*
CTPCC_ODBC_new
	( LPCSTR szServer, LPCSTR szUser,
LPCSTR szPassword, LPCSTR szHost, LPCSTR
szDatabase );

typedef CTPCC_ODBC*
(TYPE_CTPCC_ODBC)(LPCSTR, LPCSTR, LPCSTR,
LPCSTR, LPCSTR);
```

Resource.h

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

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

Tpcc.cpp

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

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

#include <sqltypes.h>

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

Appendix A:
Source Code

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

#include
"..\..\common\txnlog\include\rtetime.h"
#include
"..\..\common\txnlog\include\spinlock.h"
#include
"..\..\common\txnlog\include\txnlog.h"

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

// Txn monitor layer includes
#include
"..\..\tm_com_dll\src\tpcc_com.h"
		// COM Services
implementation on TPC-C txns
#include
"..\..\tm_tuxedo_dll\src\tpcc_tux.h"
	// interface to Tuxedo libraries
#include
"..\..\tm_encina_dll\src\tpcc_enc.h"
	// interface to Encina libraries

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



#define LEN_ERR_STRING 256

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

char
	szMyComputerName[MAX_COMPUTERNAME
_LENGTH+1];

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

// The WEBCLIENT_VERSION string specifies
the version level of this web client
interface.
// The RTE must be synchronized with the
interface level on login, otherwise the
login
// will fail.  This is a sanity check to
catch problems resulting from mismatched
versions
// of the RTE and web client.
#define WEBCLIENT_VERSION "410"

static	CRITICAL_SECTION
	TermCriticalSection;

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

TYPE_CTPCC_DBLIB	*pCTPCC_DBLIB_new;
TYPE_CTPCC_ODBC
	*pCTPCC_ODBC_new;
TYPE_CTPCC_TUXEDO
	*pCTPCC_TUXEDO_new;
TYPE_CTPCC_ENCINA	*pCTPCC_ENCINA_new;
TYPE_CTPCC_ENCINA
*pCTPCC_ENCINA_post_init;
TYPE_CTPCC_COM		*pCTPCC_COM_new;

// For deferred Delivery txns:

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

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

// configuration settings from registry
TPCCREGISTRYDATA		Reg;


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

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

/* FUNCTION: DllMain
 *
 * PURPOSE:	This function is the entry
point for the DLL.  This implementation
is based on the
```

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

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

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

       try
       {
               switch( ul_reason_for_call
)
               {
                       case
DLL_PROCESS_ATTACH:
                               {

       DWORD dwSize =
MAX_COMPUTERNAME_LENGTH+1;

       GetComputerName(szMyComputerName,
&dwSize);

       szMyComputerName[dwSize] = 0;
                               }


       DisableThreadLibraryCalls((HMODUL
E)hModule);

       InitializeCriticalSection(&TermCr
iticalSection);

                               if (
ReadTPCCRegistrySettings( &Reg ) )

       throw new CWEBCLNT_ERR(
ERR_MISSING_REGISTRY_ENTRIES );


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

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

                               TermInit();

                               // load DLL
for txn monitor
                               if
(Reg.eTxnMon == TUXEDO)
                               {

       strcpy( szDllName, Reg.szPath );

       strcat( szDllName,
"tpcc_tuxedo.dll");

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

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

                                       //
get function pointer to wrapper for class
constructor

       pCTPCC_TUXEDO_new =
(TYPE_CTPCC_TUXEDO*)
GetProcAddress(hLibInstanceTm,"CTPCC_TUXE
DO_new");
                                       if
(pCTPCC_TUXEDO_new == NULL)

       throw new CWEBCLNT_ERR(
ERR_GETPROCADDR_FAILED, szDllName,
GetLastError() );
                               }
                               else if
(Reg.eTxnMon == ENCINA)
                               {

       strcpy( szDllName, Reg.szPath );

       strcat( szDllName,
"tpcc_encina.dll");

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

       throw new CWEBCLNT_ERR(
ERR_LOADDLL_FAILED, szDllName,
GetLastError() );
                                       //
get function pointer to wrapper for class
constructor

       pCTPCC_ENCINA_new =
(TYPE_CTPCC_ENCINA*)
GetProcAddress(hLibInstanceTm,"CTPCC_ENCI
NA_new");

       pCTPCC_ENCINA_post_init =
(TYPE_CTPCC_ENCINA*)
```

```
GetProcAddress(hLibInstanceTm,"CTPCC_ENCINA_post_init");
                                        if
(pCTPCC_ENCINA_new == NULL)

        throw new CWEBCLNT_ERR(
ERR_GETPROCADDR_FAILED, szDllName,
GetLastError() );
                                }
                                else if
(Reg.eTxnMon == COM)
                                {

        strcpy( szDllName, Reg.szPath );

        strcat( szDllName,
"tpcc_com.dll");

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

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

                                        //
get function pointer to wrapper for class
constructor

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

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

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

        strcpy( szDllName, Reg.szPath );

        strcat( szDllName,
"tpcc_dblib.dll");

        hLibInstanceDb = LoadLibrary(
szDllName );

        if (hLibInstanceDb == NULL)

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


        // get function pointer to
wrapper for class constructor

        pCTPCC_DBLIB_new =
(TYPE_CTPCC_DBLIB*)
GetProcAddress(hLibInstanceDb,"CTPCC_DBLIB_new");

        if (pCTPCC_DBLIB_new == NULL)

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

        else if (Reg.eDB_Protocol ==
ODBC)
                                        {

        strcpy( szDllName, Reg.szPath );

        strcat( szDllName,
"tpcc_odbc.dll");

        hLibInstanceDb = LoadLibrary(
szDllName );

        if (hLibInstanceDb == NULL)

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


        // get function pointer to
wrapper for class constructor

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

        if (pCTPCC_ODBC_new == NULL)

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

                                if
(dwNumDeliveryThreads)
                                {
                                        //
for deferred delivery txns:

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

        InitializeCriticalSection(&DelBuffCriticalSection);

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

        dwDelBuffFreeCount =
dwDelBuffSize;


        InitJulianTime(NULL);
```

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

        SYSTEMTIME Time;

        GetLocalTime( &Time );

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

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

        txnDelilog = new
CTxnLog(szLogFile, TXN_LOG_WRITE);


        //write event into txn log for
START

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

                                        //
allocate structures for delivery buffers
and thread mgmt

        pDeliHandles = new
HANDLE[dwNumDeliveryThreads];

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

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

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

        if (pDeliHandles[i] ==
INVALID_HANDLE_VALUE)

                throw new CWEBCLNT_ERR(
ERR_DELIVERY_THREAD_FAILED );
                                        }
                                }

                                break;


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

        //write event into txn log for
STOP

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


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

        CTxnLog *txnDelilogLocal =
txnDelilog;

        txnDelilog= NULL;

        delete txnDelilogLocal;
                                        }



        delete [] pDeliHandles;

        delete [] pDelBuff;


        CloseHandle( hWorkerSemaphore );

        CloseHandle( hDoneEvent );

        DeleteCriticalSection(&DelBuffCri
ticalSection);
                                }


        DeleteCriticalSection(&TermCritic
alSection);

                                if
(hLibInstanceTm != NULL)

        FreeLibrary( hLibInstanceTm );

        hLibInstanceTm = NULL;

                                if
(hLibInstanceDb != NULL)

        FreeLibrary( hLibInstanceDb );

        hLibInstanceDb = NULL;

                                Sleep(500);
                                break;

                        default:
                                /* nothing
*/;
                }
        }
        catch (CBaseErr *e)
        {
                WriteMessageToEventLog( e-
>ErrorText() );
                delete e;
                TerminateExtension(0);
                return FALSE;
        }
        catch (...)
        {
```

```
        WriteMessageToEventLog(TEXT("Unha
ndled exception.  DLL could not load."));
                TerminateExtension(0);
                return FALSE;
        }

        return TRUE;
}


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

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

        // TODO: why do we need this here
instead of in the DLL attach?
        if (Reg.eTxnMon == ENCINA)
                pCTPCC_ENCINA_post_init();

        return TRUE;
}

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

BOOL WINAPI TerminateExtension( DWORD
dwFlags )
{
        if (pDeliHandles)
        {
                SetEvent( hDoneEvent );
                for(DWORD i=0;
i<dwNumDeliveryThreads; i++)

        WaitForSingleObject(
pDeliHandles[i], INFINITE );
        }
        TermDeleteAll();
        return TRUE;
}


/* FUNCTION: HttpExtensionProc
 *
 * PURPOSE:    This function is the main
entry point for the TPCC DLL. The
internet service
 *                     calls this
function passing in the http string.
 *
 * ARGUMENTS:  EXTENSION_CONTROL_BLOCK
        *pECB   structure pointer to
passed in internet
 *

                service information.
 *
 * RETURNS:            DWORD
        HSE_STATUS_SUCCESS
                       connection can be
dropped if error
 *
        HSE_STATUS_SUCCESS_AND_KEEP_CONN
        keep connect valid comment sent
 *
 * COMMENTS:   None
 *
 */

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

        int                     lpbSize;
        static char     szHeader[] = "200
Ok";
        DWORD           dwSize = 6;
        // initial value is
strlen(szHeader)
        char            szHeader1[4096];


#ifdef ICECAP
        StartCAP();
#endif

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

                if (TermId != 0)
                {
                        if ( TermId < 0 ||
TermId >= Term.iNumEntries ||
Term.pClientData[TermId].iNextFree != -1
)
                        {
                                //
debugging...
                                char
szTmp[128];
```

```
                              wsprintf(
szTmp, "Invalid term ID; TermId = %d",
TermId );

        WriteMessageToEventLog( szTmp );

                              throw new
CWEBCLNT_ERR( ERR_INVALID_TERMID );
                      }

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

                      //set use time

        Term.pClientData[TermId].iTickCou
nt = GetTickCount();
              }


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

        break;
                              case
NEW_ORDER_FORM:

        ProcessNewOrderForm(pECB, TermId,
szBuffer);

        break;
                              case
PAYMENT_FORM:

        ProcessPaymentForm(pECB, TermId,
szBuffer);

        break;
                              case
DELIVERY_FORM:

        ProcessDeliveryForm(pECB, TermId,
szBuffer);

        break;
                              case
ORDER_STATUS_FORM:

        ProcessOrderStatusForm(pECB,
TermId, szBuffer);

        break;
                              case
STOCK_LEVEL_FORM:

        ProcessStockLevelForm(pECB,
TermId, szBuffer);

        break;
                      }
                      break;
              case 2:
                      // new-order
selected from menu; display new-order
input form

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

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

        MakeDeliveryForm(TermId, NULL,
INPUT_FORM, szBuffer);
                      break;
              case 5:
                      // order-status
selected from menu; display order-status
input form

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

        MakeStockLevelForm(TermId, NULL,
INPUT_FORM, szBuffer);
                      break;
              case 7:
                      // ExitCmd

        TermDelete(TermId);
                      WelcomeForm(pECB,
szBuffer);
                      break;
              case 8:
                      SubmitCmd(pECB,
szBuffer);
                      break;
              case 9:
                      // menu

        MakeMainMenuForm(TermId,
Term.pClientData[TermId].iSyncId,
szBuffer);
                      break;
              case 10:
                      // CMD=Clear
                      // resets all
connections; should only be used when no
other connections are active
```

```
                        TermDeleteAll();
                        TermInit();
                        WelcomeForm(pECB,
szBuffer);
                        break;
                case 11:        //
CMD=Stats
                        StatsCmd(pECB,
szBuffer);
                        break;
                }
        }
        catch (CBaseErr *e)
        {
                ErrorForm( pECB, e-
>ErrorType(), e->ErrorNum(), TermId,
iSyncId, e->ErrorText(), szBuffer );
                delete e;
        }
        catch (...)
        {
                ErrorForm( pECB,
ERR_TYPE_WEBDLL, 0, TermId, iSyncId,
"Error: Unhandled exception in Web
Client.", szBuffer );
        }

#ifdef ICECAP
        StopCAP();
#endif


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

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

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


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

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

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

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

        (VOID)
DeregisterEventSource(hEventSource);
    }
}


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

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

        DELIVERY_TRANSACTION
        delivery;
        PDELIVERY_DATA
        pDeliveryData;
        TXN_RECORD_TPCC_DELIV_DEF
        txnDeliRec;

        DWORD
        index;
        HANDLE
        handles[2];

        SYSTEMTIME            trans_end;
                //delivery transaction
```

```
finished time

        SYSTEMTIME
        trans_start;   //delivery
transaction start time

        assert(txnDelilog != NULL);

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

        WriteMessageToEventLog(TEXT("Unha
ndled exception caught in
DeliveryWorkerThread."));
                goto ErrorExit;
        }

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

        goto ErrorExit;


        ZeroMemory(&txnDeliRec,
sizeof(txnDeliRec));

        txnDeliRec.TxnType =
TXN_REC_TYPE_TPCC_DELIV_DEF;


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

        EnterCriticalSection(&DelBuffCrit
icalSection);
                                delivery =
*(pDelBuff+dwDelBuffBusyIndex);

        dwDelBuffFreeCount++;

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

        dwDelBuffBusyIndex = 0;


        LeaveCriticalSection(&DelBuffCrit
icalSection);


        pDeliveryData->w_id =
delivery.w_id;

        pDeliveryData->o_carrier_id =
delivery.o_carrier_id;


        txnDeliRec.w_id = pDeliveryData-
>w_id;

        txnDeliRec.o_carrier_id =
pDeliveryData->o_carrier_id;

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


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

        GetLocalTime( &trans_end );

                                //log txn

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

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

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

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

                                if
(txnDelilog != NULL)

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

        WriteMessageToEventLog( szTmp );

                        // log the error
txn

        txnDeliRec.TxnStatus = e-
>ErrorType();
                        if (txnDelilog !=
NULL)
                                txnDelilog-
>WriteToLog(&txnDeliRec);

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

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

ErrorExit:
        delete pTxn;
        _endthread();
}

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

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

        EnterCriticalSection(&DelBuffCrit
icalSection);
        if (dwDelBuffFreeCount > 0)
        {
                bError = FALSE;

        (pDelBuff+dwDelBuffFreeIndex)-
>w_id                 = w_id;

        (pDelBuff+dwDelBuffFreeIndex)-
>o_carrier_id  = o_carrier_id;

        GetLocalTime(&(pDelBuff+dwDelBuff
FreeIndex)->queue);

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

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

        return bError;

}

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

void
ProcessQueryString(EXTENSION_CONTROL_BLOC
K *pECB, int *pCmd, int *pFormId, int
```

```
*pTermId, int *pSyncId)

{
	char *ptr = pECB->lpszQueryString;
	char szBuffer[25];
	int i;

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

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

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

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

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

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

/* FUNCTION: void WelcomeForm
 *
 */

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

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

	"<B><BIG>Microsoft TPC-C Web Client (ver 4.20)</BIG></B> <BR> <BR>"

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

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

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

	"</PRE></font>"

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

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

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

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

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

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

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

				);

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

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

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

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

	"Max Pending Deliveries = <B>%d</B><BR>"
				,
szTxnMonNames[Reg.eTxnMon], szDBNames[Reg.eDB_Protocol],

	Reg.dwMaxConnections, dwNumDeliveryThreads, dwDelBuffSize );
	strcat( szBuffer, szTmp);
```

```
	if (Reg.eTxnMon == COM)
	{
		sprintf( szTmp,
	"COM Single Pool       =
<B>%s</B><BR>",

	Reg.bCOM_SinglePool ? "YES" :
"NO" );
		strcat( szBuffer, szTmp);
	}
	strcat( szBuffer,
"</PRE></font>");

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

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

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

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

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

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

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

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

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

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

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

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

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

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

	"<font face=\"Courier New\"
color=\"blue\"><PRE>" );
	strcat( szBuffer, szTmp);
	strcat( szBuffer,     "Warehouse
ID = <INPUT NAME=\"w_id\" SIZE=4><BR>"

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

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

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

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

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

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

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

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

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

```
		GetKeyValue(&ptr,
"db_passwd", szPassword,
sizeof(szPassword), NO_ERR);
		// parse Database name
		GetKeyValue(&ptr,
"db_name", szDatabase,
sizeof(szDatabase), NO_ERR);
	}

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

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


	iNewTerm = TermAdd();

	Term.pClientData[iNewTerm].w_id =
w_id;
	Term.pClientData[iNewTerm].d_id =
d_id;

	try
	{
		if (Reg.eTxnMon == TUXEDO)

	Term.pClientData[iNewTerm].pTxn =
pCTPCC_TUXEDO_new();
		else if (Reg.eTxnMon ==
ENCINA)

	Term.pClientData[iNewTerm].pTxn =
pCTPCC_ENCINA_new();
		else if (Reg.eTxnMon ==
COM)

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

	Term.pClientData[iNewTerm].pTxn =
pCTPCC_ODBC_new( szServer, szUser,
szPassword, szMyComputerName, szDatabase
);
		else if (Reg.eDB_Protocol
== DBLIB)

	Term.pClientData[iNewTerm].pTxn =
pCTPCC_DBLIB_new( szServer, szUser,
szPassword, szMyComputerName, szDatabase
);
	}
	catch (...)
	{
		TermDelete(iNewTerm);
		throw;		// pass
exception upward
	}

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

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

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

	EnterCriticalSection(&TermCritica
lSection);

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

	LeaveCriticalSection(&TermCritica
lSection);

	wsprintf( szBuffer,

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

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

			},
		{	ERR_D_ID_INVALID,

	"Invalid District ID Must be 1 to
10."
			},
		{
	ERR_DELIVERY_CARRIER_ID_RANGE,
			"Delivery Carrier
ID out of range must be 1 - 10."
				},
		{
	ERR_DELIVERY_CARRIER_INVALID,
		"Delivery Carrier ID
```

```
invalid must be numeric 1 - 10."
                },
        {
        ERR_DELIVERY_MISSING_OCD_KEY,
                "Delivery missing Carrier
ID key \"OCD*\"."
                        },
        {
        ERR_DELIVERY_THREAD_FAILED,
                "Could not start
delivery worker thread."
                                },
        {
        ERR_GETPROCADDR_FAILED,
                        "Could not
map proc in DLL.  GetProcAddr error.
DLL="                   },
        {
        ERR_HTML_ILL_FORMED,
                "Required key
field is missing from HTML string."
                        },
        {
        ERR_INVALID_SYNC_CONNECTION,
                "Invalid Terminal Sync
ID."

        },
        {
        ERR_INVALID_TERMID,
                        "Invalid
Terminal ID."

                },
        {
        ERR_LOADDLL_FAILED,
                        "Load of
DLL failed.  DLL="

                },
        {
        ERR_MAX_CONNECTIONS_EXCEEDED,
                "No connections available.
Max Connections is probably too low."
},
        {
        ERR_MISSING_REGISTRY_ENTRIES,
                "Required registry entries
are missing.  Rerun INSTALL to correct."
        },
        {
        ERR_NEWORDER_CUSTOMER_INVALID,
                "New Order
customer id invalid data type, range = 1
to 3000."       },
        {
        ERR_NEWORDER_CUSTOMER_KEY,
                "New Order missing
Customer key \"CID*\"."
                                },
        {
        ERR_NEWORDER_DISTRICT_INVALID,
                "New Order
District ID Invalid range 1 - 10."
                                },
        {
        ERR_NEWORDER_FORM_MISSING_DID,
                "New Order missing
District key \"DID*\"."
                                },
        {
        ERR_NEWORDER_ITEMID_INVALID,
                "New Order Item Id is
wrong data type, must be numeric."
                },
                {
        ERR_NEWORDER_ITEMID_RANGE,
                        "New Order Item Id
is out of range. Range = 1 to 999999."
                        },
                {
        ERR_NEWORDER_ITEMID_WITHOUT_SUPPW
,               "New Order Item_Id field
entered without a corresponding Supp_W."
        },
                {
        ERR_NEWORDER_MISSING_IID_KEY,
                "New Order missing Item Id
key \"IID*\"."
                        },
                {
        ERR_NEWORDER_MISSING_QTY_KEY,
                "New Order Missing Qty key
\"Qty##*\"."
                                },
                {
        ERR_NEWORDER_MISSING_SUPPW_KEY,
                        "New Order missing
Supp_W key \"SP##*\"."
                                },
                {
        ERR_NEWORDER_NOITEMS_ENTERED,
                "New Order No order lines
entered."
                                        },
                {
        ERR_NEWORDER_QTY_INVALID,
                        "New Order Qty
invalid must be numeric range 1 - 99."
                                },
                {
        ERR_NEWORDER_QTY_RANGE,
                                "New Order
Qty is out of range. Range = 1 to 99."
                                        },
                {
        ERR_NEWORDER_QTY_WITHOUT_SUPPW,
                        "New Order Qty
field entered without a corresponding
Supp_W."                },
                {
        ERR_NEWORDER_SUPPW_INVALID,
                        "New Order Supp_W
invalid data type must be numeric."
                        },
                {
        ERR_NO_SERVER_SPECIFIED,
                        "No Server name
specified."

                },
                {
        ERR_ORDERSTATUS_CID_AND_CLT,
                "Order Status Only
Customer ID or Last Name may be entered,
not both."      },
                {
        ERR_ORDERSTATUS_CID_INVALID,
                "Order Status Customer ID
```

```
invalid, range must be numeric 1 - 3000."
        },
                {
        ERR_ORDERSTATUS_CLT_RANGE,
                        "Order Status
Customer last name longer than 16
characters."            },
                {
        ERR_ORDERSTATUS_DID_INVALID,
                "Order Status District
invalid, value must be numeric 1 - 10."
                },
                {
        ERR_ORDERSTATUS_MISSING_CID_CLT,
                "Order Status Either
Customer ID or Last Name must be
entered."               },
                {
        ERR_ORDERSTATUS_MISSING_CID_KEY,
                "Order Status missing
Customer key \"CID*\"."
                                },
                {
        ERR_ORDERSTATUS_MISSING_CLT_KEY,
                "Order Status missing
Customer Last Name key \"CLT*\"."
                        },
                {
        ERR_ORDERSTATUS_MISSING_DID_KEY,
                "Order Status missing
District key \"DID*\"."
                                },
                {
        ERR_PAYMENT_CDI_INVALID,
                        "Payment Customer
district invalid must be numeric."
                        },
                {
        ERR_PAYMENT_CID_AND_CLT,
                        "Payment Only
Customer ID or Last Name may be entered,
not both."      },
                {
        ERR_PAYMENT_CUSTOMER_INVALID,
                "Payment Customer data
type invalid, must be numeric."
                        },
                {
        ERR_PAYMENT_CWI_INVALID,
                        "Payment Customer
Warehouse invalid, must be numeric."
                        },
                {
        ERR_PAYMENT_DISTRICT_INVALID,
                "Payment District ID is
invalid, must be 1 - 10."
                        },
                {
        ERR_PAYMENT_HAM_INVALID,
                        "Payment Amount
invalid data type must be numeric."
                                },
                {
        ERR_PAYMENT_HAM_RANGE,
                        "Payment Amount
out of range, 0 - 9999.99."
                                        },
                {
        ERR_PAYMENT_LAST_NAME_TO_LONG,
                        "Payment Customer
last name longer than 16 characters."
                        },
                {
        ERR_PAYMENT_MISSING_CDI_KEY,
                "Payment missing Customer
district key \"CDI*\"."
                        },
                {
        ERR_PAYMENT_MISSING_CID_CLT,
                "Payment Either Customer
ID or Last Name must be entered."
                },
                {
        ERR_PAYMENT_MISSING_CID_KEY,
                "Payment missing Customer
Key \"CID*\"."
                        },
                {
        ERR_PAYMENT_MISSING_CLT_KEY,
                "Payment missing Customer
Last Name key \"CLT*\"."
                        },
                {
        ERR_PAYMENT_MISSING_CWI_KEY,
                "Payment missing Customer
Warehouse key \"CWI*\"."
                        },
                {
        ERR_PAYMENT_MISSING_DID_KEY,
                "Payment missing District
Key \"DID*\"."
                        },
                {
        ERR_PAYMENT_MISSING_HAM_KEY,
                "Payment missing Amount
key \"HAM*\"."
                                },
                {
        ERR_STOCKLEVEL_MISSING_THRESHOLD_
KEY,    "Stock Level; missing Threshold
key \"TT*\"."
                },
                {
        ERR_STOCKLEVEL_THRESHOLD_INVALID,
                "Stock Level; Threshold
value must be in the range = 1 - 99."
        },
                {
        ERR_STOCKLEVEL_THRESHOLD_RANGE,
                        "Stock Level
Threshold out of range, range must be 1 -
99."                    },
                {
        ERR_VERSION_MISMATCH,
                        "Invalid version
field.  RTE and Web Client are probably
out of sync." },
                {       ERR_W_ID_INVALID,

        "Invalid Warehouse ID."


        },
                {       0,

                        ""


                                        }
        };
```

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

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

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

}

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

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

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

	iMax--;  // one position is for
terminating null
	while( *ptr && *ptr != '&' &&
iMax)
	{
		*pValue++ = *ptr++;
		iMax--;
	}
	*pValue = 0;  // terminating null

	*pQueryString = ptr;
	return;

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

/* FUNCTION: GetIntKeyValue
 *
 * PURPOSE:    This function parses a
http formatted string for a specific key
value.
 *
 * ARGUMENTS:  char
	*pQueryString  http string from
client browser
 *				char
		*pKey			key
value to look for
 *				WEBERROR
		NoKeyErr
	error value to throw if key not
found
 *				WEBERROR
		NotIntErr
	error value to throw if value not
numeric
```

Appendix A:
Source Code

```
 *
 * RETURNS:           integer
 *
 * ERROR:             if (the pKey value
is not found) then
 *                                   if
(NoKeyErr != NO_ERR)
 *
       throw CWEBCLNT_ERR(err)
 *
       else
 *
       return 0
 *                            else if
(non-numeric char found) then
 *                                   if
(NotIntErr != NO_ERR) then
 *
       throw CWEBCLNT_ERR(err)
 *
       else
 *
       return 0
 *
 * COMMENTS:   http keys are formatted
either KEY=value& or KEY=value\0. This
DLL formats
 *                            TPC-C input
fields in such a manner that the keys can
be extracted in the
 *                            above
manner.
 */

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

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

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

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

       *pQueryString = ptr;
       return atoi(ptr0);

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

       return 0;
}

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

void TermInit(void)
{
       EnterCriticalSection(&TermCritica
lSection);

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

       Term.pClientData      = NULL;
       Term.pClientData      =
(PCLIENTDATA)malloc(Term.iNumEntries *
sizeof(CLIENTDATA));
       if (Term.pClientData == NULL)
       {

       LeaveCriticalSection(&TermCritica
lSection);
              throw new CWEBCLNT_ERR(
ERR_MEM_ALLOC_FAILED );
       }

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

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

       Term.pClientData[i].iNextFree =
i-1;

       LeaveCriticalSection(&TermCritica
lSection);
}

/* FUNCTION: TermDeleteAll
 *
 * PURPOSE:    This function frees
allocated resources associated with the
terminal structure.
```

```
 *
 * ARGUMENTS:  none
 *
 * RETURNS:            None
 *
 * COMMENTS:   This function is called
only when the inet service unloads the
TPCC.DLL
 *
 */

void TermDeleteAll(void)
{
       EnterCriticalSection(&TermCritica
lSection);

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

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

       LeaveCriticalSection(&TermCritica
lSection);
}

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

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

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

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

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

       LeaveCriticalSection(&TermCritica
lSection);
                     throw new
CWEBCLNT_ERR(
ERR_MAX_CONNECTIONS_EXCEEDED );
              }
       }

       Term.pClientData[iNewTerm].iTickC
ount = GetTickCount();
       Term.pClientData[iNewTerm].iSyncI
d = Term.iMasterSyncId++;
       Term.pClientData[iNewTerm].pTxn =
NULL;

       LeaveCriticalSection(&TermCritica
lSection);
       return iNewTerm;
}

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

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

              // put onto free list
```

```
	EnterCriticalSection(&TermCritica
lSection);


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


	LeaveCriticalSection(&TermCritica
lSection);
	}
}


/* FUNCTION: MakeErrorForm
 */

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

/* FUNCTION: MakeMainMenuForm
 */

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

/* FUNCTION: MakeStockLevelForm
 *
 * PURPOSE:	This function constructs
the Stock Level HTML page.
 *
 * COMMENTS:	The internal client buffer
is created when the terminal id is
assigned and should not
 *				be freed
except when the client terminal id is no
longer needed.
 */

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

	c = wsprintf(szForm,
		"<HTML><HEAD><TITLE>TPC-C
Stock Level</TITLE></HEAD><FORM
ACTION=\"tpcc.dll\" METHOD=\"GET\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"0\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
		"<PRE><font
face=\"Courier\">
Stock-Level<BR>"
		"Warehouse: %4.4d
District: %2.2d<BR> <BR>",
```

```
		STOCK_LEVEL_FORM, iTermId,
Term.pClientData[iTermId].iSyncId,

	Term.pClientData[iTermId].w_id,
Term.pClientData[iTermId].d_id);

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

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

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

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

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

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

	if ( bInput )
	{
		c += wsprintf(szForm+c,
"Warehouse: %4.4d   ",
Term.pClientData[iTermId].w_id );


		strcpy( szForm+c,
			"District: <INPUT
NAME=\"DID*\" SIZE=1>
Date:<BR>"
			"Customer:  <INPUT
NAME=\"CID*\" SIZE=4>   Name:
Credit:      %Disc:<BR>"
			"Order Number:
Number of Lines:           W_tax:
D_tax:<BR> <BR>"
			" Supp_W  Item_Id
Item Name                 Qty  Stock  B/G
Price    Amount<BR>"
			"  <INPUT
NAME=\"SP00*\" SIZE=4>  <INPUT
NAME=\"IID00*\" SIZE=6>
<INPUT NAME=\"Qty00*\" SIZE=1><BR>"
			"  <INPUT
NAME=\"SP01*\" SIZE=4>  <INPUT
NAME=\"IID01*\" SIZE=6>
<INPUT NAME=\"Qty01*\" SIZE=1><BR>"
```

```
                        "  <INPUT
NAME=\"SP02*\" SIZE=4>  <INPUT
NAME=\"IID02*\" SIZE=6>
<INPUT NAME=\"Qty02*\" SIZE=1><BR>"

                        "  <INPUT
NAME=\"SP03*\" SIZE=4>  <INPUT
NAME=\"IID03*\" SIZE=6>
<INPUT NAME=\"Qty03*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP04*\" SIZE=4>  <INPUT
NAME=\"IID04*\" SIZE=6>
<INPUT NAME=\"Qty04*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP05*\" SIZE=4>  <INPUT
NAME=\"IID05*\" SIZE=6>
<INPUT NAME=\"Qty05*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP06*\" SIZE=4>  <INPUT
NAME=\"IID06*\" SIZE=6>
<INPUT NAME=\"Qty06*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP07*\" SIZE=4>  <INPUT
NAME=\"IID07*\" SIZE=6>
<INPUT NAME=\"Qty07*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP08*\" SIZE=4>  <INPUT
NAME=\"IID08*\" SIZE=6>
<INPUT NAME=\"Qty08*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP09*\" SIZE=4>  <INPUT
NAME=\"IID09*\" SIZE=6>
<INPUT NAME=\"Qty09*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP10*\" SIZE=4>  <INPUT
NAME=\"IID10*\" SIZE=6>
<INPUT NAME=\"Qty10*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP11*\" SIZE=4>  <INPUT
NAME=\"IID11*\" SIZE=6>
<INPUT NAME=\"Qty11*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP12*\" SIZE=4>  <INPUT
NAME=\"IID12*\" SIZE=6>
<INPUT NAME=\"Qty12*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP13*\" SIZE=4>  <INPUT
NAME=\"IID13*\" SIZE=6>
<INPUT NAME=\"Qty13*\" SIZE=1><BR>"
                        "  <INPUT
NAME=\"SP14*\" SIZE=4>  <INPUT
NAME=\"IID14*\" SIZE=6>
<INPUT NAME=\"Qty14*\" SIZE=1><BR>"
                        "Execution Status:
Total:<BR>"

        "</font></PRE><HR>"
                        "<INPUT
TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"Process\">"
                        "<INPUT
TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"Menu\">"
                        "</FORM></HTML>"
                        );
        }
        else
        {
                        c += wsprintf(szForm+c,
"Warehouse: %4.4d   District: %2.2d
Date: ",
                                pNewOrderData-
>w_id,
                                pNewOrderData-
>d_id);

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

        pNewOrderData->o_entry_d.day,

        pNewOrderData->o_entry_d.month,

        pNewOrderData->o_entry_d.year,

        pNewOrderData->o_entry_d.hour,

        pNewOrderData->o_entry_d.minute,

        pNewOrderData->o_entry_d.second);
                        }

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

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

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

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

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

        100.0*pNewOrderData->c_discount,

        pNewOrderData->o_id,


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

                                for(i=0;
i<pNewOrderData->o_ol_cnt; i++)
                                {
                                        c +=
sprintf(szForm+c, "  %4.4d   %6.6d   %-
24s  %2.2d    %3.3d    %1.1s   $%6.2f
$%7.2f  <BR>",

        pNewOrderData-
>OL[i].ol_supply_w_id,
```

Appendix A:
Source Code

```
	pNewOrderData->OL[i].ol_i_id,

	pNewOrderData->OL[i].ol_i_name,

	pNewOrderData->OL[i].ol_quantity,

	pNewOrderData->OL[i].ol_stock,

	pNewOrderData-
>OL[i].ol_brand_generic,

	pNewOrderData->OL[i].ol_i_price,

	pNewOrderData->OL[i].ol_amount );
				}
			}
			else
			{
				c +=
wsprintf(szForm+c,

	"%Disc:<BR>"
					"Order
Number: %8.8d  Number of Lines:
W_tax:         D_tax:<BR> <BR>"
					" Supp_W
Item_Id  Item Name                 Qty
Stock  B/G  Price    Amount<BR>"
					,
pNewOrderData->o_id);

				i = 0;
			}

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

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

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

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

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

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

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

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

	if ( bInput )
	{
		c += wsprintf(szForm+c,
			"<BR>
<BR>Warehouse: %4.4d"
```

```
                        "
District: <INPUT NAME=\"DID*\"
SIZE=1><BR> <BR> <BR> <BR> <BR>"
                        "Customer: <INPUT
NAME=\"CID*\" SIZE=4>"
                        "Cust-Warehouse:
<INPUT NAME=\"CWI*\" SIZE=4>  "
                        "Cust-District:
<INPUT NAME=\"CDI*\" SIZE=1><BR>"
                        "Name:
<INPUT NAME=\"CLT*\" SIZE=16>
Since:<BR>"
                        "
Credit:<BR>"
                        "
Disc:<BR>"
                        "
Phone:<BR> <BR>"
                        "Amount Paid:
$<INPUT NAME=\"HAM*\" SIZE=7>      New
Cust-Balance:<BR>"
                        "Credit Limit:<BR>
<BR>Cust-Data: <BR> <BR> <BR> <BR>
<BR></font></PRE><HR>"
                        "<INPUT
TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"Process\"><INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Menu\">"

        "</BODY></FORM></HTML>"
                        ,
Term.pClientData[iTermId].w_id);
        }
        else
        {
                c += wsprintf(szForm+c,
                        "<BR>
<BR>Warehouse: %4.4d
District: %2.2d<BR>"
                        "%-20s
%-20s<BR>"
                        "%-20s
%-20s<BR>"
                        "%-20s %-2s %5.5s-
%4.4s       %-20s %-2s %5.5s-%4.4s<BR>
<BR>"
                        "Customer: %4.4d
Cust-Warehouse: %4.4d  Cust-District:
%2.2d<BR>"
                        "Name:   %-16s %-
2s %-16s     Since:  %2.2d-%2.2d-
%4.4d<BR>"
                        "       %-20s
Credit: %-2s<BR>"


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

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


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

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

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

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

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

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

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

Appendix A:
Source Code

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

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

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

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

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

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

}

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

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

	c = wsprintf(szForm,
		"<HTML><HEAD><TITLE>TPC-C
Order-Status</TITLE></HEAD><BODY>"
		"<FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"0\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
		"<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
		"<PRE><font
face=\"Courier\">
Order-Status<BR>"
		"Warehouse: %4.4d   ",
		ORDER_STATUS_FORM,
iTermId,
Term.pClientData[iTermId].iSyncId,
Term.pClientData[iTermId].w_id);

	if ( bInput )
	{
		strcpy(szForm+c,
			"District: <INPUT
NAME=\"DID*\" SIZE=1><BR>"
			"Customer: <INPUT
NAME=\"CID*\" SIZE=4>   Name:
<INPUT NAME=\"CLT*\" SIZE=23><BR>"
			"Cust-Balance:<BR>
<BR>"
			"Order-Number:
Entry-Date:
Carrier-Number:<BR>"
			"Supply-W
Item-Id    Qty     Amount      Delivery-
Date<BR> <BR> <BR> <BR> <BR>"
			" <BR> <BR> <BR>
<BR> <BR> <BR> <BR> <BR> <BR> <BR>
<BR></font></PRE>"
			"<HR><INPUT
TYPE=\"submit\" NAME=\"CMD\"
VALUE=\"Process\"><INPUT TYPE=\"submit\"
NAME=\"CMD\" VALUE=\"Menu\">"

	"</BODY></FORM></HTML>" );
	}
	else
	{
		c += wsprintf(szForm+c,
			"District:
%2.2d<BR>"
			"Customer: %4.4d
Name: %-16s %-2s %-16s<BR>",
			pOrderStatusData-
>d_id, pOrderStatusData->c_id,
			pOrderStatusData-
>c_first, pOrderStatusData->c_middle,
pOrderStatusData->c_last);

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

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

		for(i=0; i<
pOrderStatusData->o_ol_cnt; i++)
		{
			c +=
sprintf(szForm+c, "  %4.4d       %6.6d
```

```
%2.2d     $%8.2f      %2.2d-%2.2d-
%4.4d<BR>",

        pOrderStatusData-
>OL[i].ol_supply_w_id,

        pOrderStatusData->OL[i].ol_i_id,

        pOrderStatusData-
>OL[i].ol_quantity,

        pOrderStatusData-
>OL[i].ol_amount,

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

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

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

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

                strcpy(szForm+c,

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

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

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

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

        c = wsprintf(szForm,
                "<HTML><HEAD><TITLE>TPC-C
Delivery</TITLE></HEAD><BODY>"
                "<FORM ACTION=\"tpcc.dll\"
METHOD=\"GET\">"
                "<INPUT TYPE=\"hidden\"
NAME=\"STATUSID\" VALUE=\"%d\">"
                "<INPUT TYPE=\"hidden\"
NAME=\"ERROR\" VALUE=\"0\">"
                "<INPUT TYPE=\"hidden\"
NAME=\"FORMID\" VALUE=\"%d\">"
                "<INPUT TYPE=\"hidden\"
NAME=\"TERMID\" VALUE=\"%d\">"
                "<INPUT TYPE=\"hidden\"
NAME=\"SYNCID\" VALUE=\"%d\">"
                "<PRE><font
face=\"Courier\">
Delivery<BR>"
                "Warehouse: %4.4d<BR>
<BR>",
                (!bInput &&
(pDeliveryData->exec_status_code != eOK))
? ERR_TYPE_DELIVERY_POST : 0,
                DELIVERY_FORM, iTermId,
Term.pClientData[iTermId].iSyncId,
Term.pClientData[iTermId].w_id);

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

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

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

	"</BODY></FORM></HTML>"

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

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

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

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

	ZeroMemory(pNewOrder,
sizeof(NEW_ORDER_DATA));
	pNewOrder->w_id =
Term.pClientData[iTermId].w_id;
	GetNewOrderData(pECB-
>lpszQueryString, pNewOrder);

	Term.pClientData[iTermId].pTxn-
>NewOrder();

	pNewOrder =
Term.pClientData[iTermId].pTxn-
>BuffAddr_NewOrder();
	MakeNewOrderForm(iTermId,
pNewOrder, OUTPUT_FORM, szBuffer );
}

/* FUNCTION: void ProcessPaymentForm
 *
 * PURPOSE:	This function gets and
validates the input data from the payment
form
 *			filling in the
required input variables. It then calls
the SQLPayment
 *			transaction,
constructs the output form and writes it
back to client
 *			browser.
 *
 * ARGUMENTS:	EXTENSION_CONTROL_BLOCK
	*pECB	passed in structure
pointer from inetsrv.
 *				int

	iTermId	client browser terminal id
 *
 */

void
ProcessPaymentForm(EXTENSION_CONTROL_BLOC
K *pECB, int iTermId, char *szBuffer)
{
	PPAYMENT_DATA	pPayment;

	pPayment =
Term.pClientData[iTermId].pTxn-
>BuffAddr_Payment();
	ZeroMemory(pPayment,
sizeof(PAYMENT_DATA));
	pPayment->w_id =
Term.pClientData[iTermId].w_id;
	GetPaymentData(pECB-
>lpszQueryString, pPayment);

	Term.pClientData[iTermId].pTxn-
>Payment();

	pPayment =
Term.pClientData[iTermId].pTxn-
>BuffAddr_Payment();
	MakePaymentForm(iTermId,
pPayment, OUTPUT_FORM, szBuffer);
}

/* FUNCTION: ProcessOrderStatusForm
 *
 * PURPOSE:	This function gets and
validates the input data from the Order
Status
 *			form filling in
the required input variables. It then
calls the
 *			SQLOrderStatus
transaction, constructs the output form
and writes it
 *			back to client
browser.
 *
 * ARGUMENTS:	EXTENSION_CONTROL_BLOCK
	*pECB	passed in structure
pointer from inetsrv.
 *				int

	iTermId	client browser terminal id
 *
 */

void
ProcessOrderStatusForm(EXTENSION_CONTROL_
BLOCK *pECB, int iTermId, char *szBuffer)
{
	PORDER_STATUS_DATA
	pOrderStatus;

	pOrderStatus =
Term.pClientData[iTermId].pTxn-
>BuffAddr_OrderStatus();
```

```
	ZeroMemory(pOrderStatus, 
sizeof(ORDER_STATUS_DATA));
	pOrderStatus->w_id = 
Term.pClientData[iTermId].w_id;
	GetOrderStatusData(pECB-
>lpszQueryString, pOrderStatus);

	Term.pClientData[iTermId].pTxn-
>OrderStatus();

	pOrderStatus = 
Term.pClientData[iTermId].pTxn-
>BuffAddr_OrderStatus();
	MakeOrderStatusForm(iTermId, 
pOrderStatus, OUTPUT_FORM, szBuffer);
}

/* FUNCTION: ProcessDeliveryForm
 *
 * PURPOSE:	This function gets and 
validates the input data from the 
delivery form
 *			filling in the 
required input variables. It then calls 
the PostDeliveryInfo
 *			Api, The client is 
then informed that the transaction has 
been posted.
 *
 * ARGUMENTS:  EXTENSION_CONTROL_BLOCK	
	*pECB	passed in structure 
pointer from inetsrv.
 *				int		
	iTermId	client browser terminal id
 *
 */

void 
ProcessDeliveryForm(EXTENSION_CONTROL_BLO
CK *pECB, int iTermId, char *szBuffer)
{
	char	*ptr = pECB-
>lpszQueryString;

	PDELIVERY_DATA pDelivery;

	pDelivery = 
Term.pClientData[iTermId].pTxn-
>BuffAddr_Delivery();
	ZeroMemory(pDelivery, 
sizeof(DELIVERY_DATA));
	pDelivery->w_id = 
Term.pClientData[iTermId].w_id;

	pDelivery->o_carrier_id	= 
GetIntKeyValue(&ptr, "OCD*", 
ERR_DELIVERY_MISSING_OCD_KEY, 
ERR_DELIVERY_CARRIER_INVALID);
	if ( pDelivery->o_carrier_id > 10 
|| pDelivery->o_carrier_id < 1 )
		throw new CWEBCLNT_ERR( 
ERR_DELIVERY_CARRIER_ID_RANGE );

	if (dwNumDeliveryThreads)
	{
		//post delivery info
		if ( 
PostDeliveryInfo(pDelivery->w_id, 
pDelivery->o_carrier_id) )
			pDelivery-
>exec_status_code = eDeliveryFailed;
		else
			pDelivery-
>exec_status_code = eOK;
	}
	else // delivery is done 
synchronously if no delivery threads 
configured
	{
	Term.pClientData[iTermId].pTxn-
>Delivery();

	pDelivery = 
Term.pClientData[iTermId].pTxn-
>BuffAddr_Delivery();
	MakeDeliveryForm(iTermId, 
pDelivery, OUTPUT_FORM, szBuffer);
}

/* FUNCTION: ProcessStockLevelForm
 *
 * PURPOSE:	This function gets and 
validates the input data from the Stock 
Level
 *			form filling in 
the required input variables. It then 
calls the
 *			SQLStockLevel 
transaction, constructs the output form 
and writes it
 *			back to client 
browser.
 *
 * ARGUMENTS:  EXTENSION_CONTROL_BLOCK	
	*pECB	passed in structure 
pointer from inetsrv.
 *				int		
	iTermId	client browser terminal id
 *
 */

void 
ProcessStockLevelForm(EXTENSION_CONTROL_B
LOCK *pECB, int iTermId, char *szBuffer)
{
	char		*ptr = pECB-
>lpszQueryString;

	PSTOCK_LEVEL_DATA	
	pStockLevel;

	pStockLevel = 
Term.pClientData[iTermId].pTxn-
>BuffAddr_StockLevel();
	ZeroMemory( pStockLevel, 
sizeof(STOCK_LEVEL_DATA) );

	pStockLevel->w_id = 
Term.pClientData[iTermId].w_id;
	pStockLevel->d_id = 
Term.pClientData[iTermId].d_id;

	pStockLevel->threshold = 
GetIntKeyValue(&ptr, "TT*", 
ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY, 
ERR_STOCKLEVEL_THRESHOLD_INVALID);
	if ( pStockLevel->threshold >= 
100 || pStockLevel->threshold < 0 )
```

```
		throw new CWEBCLNT_ERR( 
ERR_STOCKLEVEL_THRESHOLD_RANGE );

	Term.pClientData[iTermId].pTxn->StockLevel();

	pStockLevel = 
Term.pClientData[iTermId].pTxn->BuffAddr_StockLevel();
	MakeStockLevelForm(iTermId, 
pStockLevel, OUTPUT_FORM, szBuffer);
}

/* FUNCTION: GetNewOrderData
 *
 * PURPOSE:	This function extracts and 
validates the new order form data from an 
http command string.
 *
 * ARGUMENTS:  LPSTR
	lpszQueryString
	client browser http command 
string
 *
	NEW_ORDER_DATA *pNewOrderData
	pointer to new order data 
structure
 *
 */

void GetNewOrderData(LPSTR 
lpszQueryString, NEW_ORDER_DATA 
*pNewOrderData)
{
	char	szTmp[26];
	int		i;
	short	items;
	int		ol_i_id, 
ol_quantity;
	char	*ptr = lpszQueryString;

	static char 
szSP[MAX_OL_NEW_ORDER_ITEMS][6] =
		{ "SP00*", "SP01*", 
"SP02*", "SP03*", "SP04*",
		  "SP05*", "SP06*", 
"SP07*", "SP08*", "SP09*",
		  "SP10*", "SP11*", 
"SP12*", "SP13*", "SP14*" };
	static char 
szIID[MAX_OL_NEW_ORDER_ITEMS][7] =
		{ "IID00*", "IID01*", 
"IID02*", "IID03*", "IID04*",
		  "IID05*", "IID06*", 
"IID07*", "IID08*", "IID09*",
		  "IID10*", "IID11*", 
"IID12*", "IID13*", "IID14*" };
	static char 
szQty[MAX_OL_NEW_ORDER_ITEMS][7] =
		{ "Qty00*", "Qty01*", 
"Qty02*", "Qty03*", "Qty04*",
		  "Qty05*", "Qty06*", 
"Qty07*", "Qty08*", "Qty09*",
		  "Qty10*", "Qty11*", 
"Qty12*", "Qty13*", "Qty14*" };

	pNewOrderData->d_id = 
GetIntKeyValue(&ptr, "DID*", 
ERR_NEWORDER_FORM_MISSING_DID, 
ERR_NEWORDER_DISTRICT_INVALID);
	pNewOrderData->c_id = 
GetIntKeyValue(&ptr, "CID*", 
ERR_NEWORDER_CUSTOMER_KEY, 
ERR_NEWORDER_CUSTOMER_INVALID);

	for(i=0, items=0; 
i<MAX_OL_NEW_ORDER_ITEMS; i++)
	{
		GetKeyValue(&ptr, szSP[i], 
szTmp, sizeof(szTmp), 
ERR_NEWORDER_MISSING_SUPPW_KEY);
		if ( szTmp[0] )
		{
			if ( 
!IsNumeric(szTmp) )
				throw new 
CWEBCLNT_ERR( ERR_NEWORDER_SUPPW_INVALID 
);
			pNewOrderData->OL[items].ol_supply_w_id = 
(short)atoi(szTmp);

			ol_i_id = 
pNewOrderData->OL[items].ol_i_id = 

	GetIntKeyValue(&ptr, szIID[i], 
ERR_NEWORDER_MISSING_IID_KEY, 
ERR_NEWORDER_ITEMID_INVALID);
			if ( ol_i_id > 
999999 || ol_i_id < 1 )
				throw new 
CWEBCLNT_ERR( ERR_NEWORDER_ITEMID_RANGE 
);

			ol_quantity = 
pNewOrderData->OL[items].ol_quantity = 

	GetIntKeyValue(&ptr, szQty[i], 
ERR_NEWORDER_MISSING_QTY_KEY, 
ERR_NEWORDER_QTY_INVALID);
			if ( ol_quantity > 
99 || ol_quantity < 1 )
				throw new 
CWEBCLNT_ERR( ERR_NEWORDER_QTY_RANGE );

			items++;
		}
		else
		{	// nothing entered 
for supply warehouse, so item id and qty 
must also be blank
			GetKeyValue(&ptr, 
szIID[i], szTmp, sizeof(szTmp), 
ERR_NEWORDER_MISSING_IID_KEY);
			if ( szTmp[0] )
				throw new 
CWEBCLNT_ERR( 
ERR_NEWORDER_ITEMID_WITHOUT_SUPPW );

			GetKeyValue(&ptr, 
szQty[i], szTmp, sizeof(szTmp), 
ERR_NEWORDER_MISSING_QTY_KEY);
			if ( szTmp[0] )
				throw new 
CWEBCLNT_ERR( 
ERR_NEWORDER_QTY_WITHOUT_SUPPW );
		}
	}
	if ( items == 0 )
```

```
		throw new CWEBCLNT_ERR( 
ERR_NEWORDER_NOITEMS_ENTERED );

	pNewOrderData->o_ol_cnt = items;
}

/* FUNCTION: GetPaymentData
 *
 * PURPOSE:	This function extracts and 
validates the payment form data from an 
http command string.
 *
 * ARGUMENTS:  LPSTR
	lpszQueryString
	client browser http command 
string
 *
	PAYMENT_DATA   *pPaymentData
	pointer to payment data structure
 */

void GetPaymentData(LPSTR 
lpszQueryString, PAYMENT_DATA 
*pPaymentData)
{
	char	szTmp[26];
	char	*ptr = lpszQueryString;
	BOOL	bCustIdBlank;

	pPaymentData->d_id = 
GetIntKeyValue(&ptr, "DID*", 
ERR_PAYMENT_MISSING_DID_KEY, 
ERR_PAYMENT_DISTRICT_INVALID);

	GetKeyValue(&ptr, "CID*", szTmp, 
sizeof(szTmp), 
ERR_PAYMENT_MISSING_CID_KEY);
	if ( szTmp[0] == 0 )
	{
		bCustIdBlank = TRUE;
		pPaymentData->c_id = 0;
	}
	else
	{	// parse customer id and 
verify that last name was NOT entered
		bCustIdBlank = FALSE;
		if ( !IsNumeric(szTmp) )
			throw new 
CWEBCLNT_ERR( 
ERR_PAYMENT_CUSTOMER_INVALID );
		pPaymentData->c_id = 
atoi(szTmp);
	}

	pPaymentData->c_w_id = 
GetIntKeyValue(&ptr, "CWI*", 
ERR_PAYMENT_MISSING_CWI_KEY, 
ERR_PAYMENT_CWI_INVALID);
	pPaymentData->c_d_id = 
GetIntKeyValue(&ptr, "CDI*", 
ERR_PAYMENT_MISSING_CDI_KEY, 
ERR_PAYMENT_CDI_INVALID);

	if ( bCustIdBlank )
	{	// customer id is blank, 
so last name must be entered
		GetKeyValue(&ptr, "CLT*", 
szTmp, sizeof(szTmp), 
ERR_PAYMENT_MISSING_CLT_KEY);
		if ( szTmp[0] == 0 )
			throw new 
CWEBCLNT_ERR( ERR_PAYMENT_MISSING_CID_CLT 
);

		_strupr( szTmp );
		if ( strlen(pPaymentData-
>c_last) > LAST_NAME_LEN )
			throw new 
CWEBCLNT_ERR( 
ERR_PAYMENT_LAST_NAME_TO_LONG );
		strcpy(pPaymentData-
>c_last, szTmp);
	}
	else
	{	// parse customer id and 
verify that last name was NOT entered
		GetKeyValue(&ptr, "CLT*", 
szTmp, sizeof(szTmp), 
ERR_PAYMENT_MISSING_CLT_KEY);
		if ( szTmp[0] != 0 )
			throw new 
CWEBCLNT_ERR( ERR_PAYMENT_CID_AND_CLT );
	}

	GetKeyValue(&ptr, "HAM*", szTmp, 
sizeof(szTmp), 
ERR_PAYMENT_MISSING_HAM_KEY);
	if (!IsDecimal(szTmp))
		throw new CWEBCLNT_ERR( 
ERR_PAYMENT_HAM_INVALID );
	pPaymentData->h_amount = 
atof(szTmp);
	if ( pPaymentData->h_amount >= 
10000.00 || pPaymentData->h_amount < 0 )
		throw new CWEBCLNT_ERR( 
ERR_PAYMENT_HAM_RANGE );
}

/* FUNCTION: GetOrderStatusData
 *
 * PURPOSE:	This function extracts and 
validates the payment form data from an 
http command string.
 *
 */
void GetOrderStatusData(LPSTR 
lpszQueryString, ORDER_STATUS_DATA 
*pOrderStatusData)
{
	char	szTmp[26];
	char	*ptr = lpszQueryString;

	pOrderStatusData->d_id = 
GetIntKeyValue(&ptr, "DID*", 
ERR_ORDERSTATUS_MISSING_DID_KEY, 
ERR_ORDERSTATUS_DID_INVALID);

	GetKeyValue(&ptr, "CID*", szTmp, 
sizeof(szTmp), 
ERR_ORDERSTATUS_MISSING_CID_KEY);
	if ( szTmp[0] == 0 )
	{	// customer id is blank, 
so last name must be entered
		pOrderStatusData->c_id = 
0;
		GetKeyValue(&ptr, "CLT*", 
szTmp, sizeof(szTmp), 
ERR_ORDERSTATUS_MISSING_CLT_KEY);
		if ( szTmp[0] == 0 )
```

```
                        throw new
CWEBCLNT_ERR(
ERR_ORDERSTATUS_MISSING_CID_CLT );

                _strupr( szTmp );
                if (
strlen(pOrderStatusData->c_last) >
LAST_NAME_LEN )
                        throw new
CWEBCLNT_ERR( ERR_ORDERSTATUS_CLT_RANGE
);
                strcpy(pOrderStatusData-
>c_last, szTmp);
        }
        else
        {       // parse customer id and
verify that last name was NOT entered
                if ( !IsNumeric(szTmp) )
                        throw new
CWEBCLNT_ERR( ERR_ORDERSTATUS_CID_INVALID
);
                pOrderStatusData->c_id =
atoi(szTmp);
                GetKeyValue(&ptr, "CLT*",
szTmp, sizeof(szTmp),
ERR_ORDERSTATUS_MISSING_CLT_KEY);
                if ( szTmp[0] != 0 )
                        throw new
CWEBCLNT_ERR( ERR_ORDERSTATUS_CID_AND_CLT
);
        }

}

/* FUNCTION: BOOL IsNumeric(char *ptr)
 *
 * PURPOSE:     This function determines
if a string is numeric. It fails if any
characters other
 *                      than numeric and
null terminator are present.
 *
 * ARGUMENTS:   char
        *ptr    pointer to string to
check.
 *
 * RETURNS:             BOOL    FALSE   if
string is not all numeric
 *
        TRUE    if string contains only
numeric characters i.e. '0' - '9'
 */

BOOL IsNumeric(char *ptr)
{
        if ( *ptr == 0 )
                return FALSE;

        while( *ptr && isdigit(*ptr) )
                ptr++;
        return ( !*ptr );
}

/* FUNCTION: BOOL IsDecimal(char *ptr)
 *
 * PURPOSE:     This function determines
if a string is a non-negative decimal
value.
 *              It fails if any characters
other than a series of numbers followed
by
 *                      a decimal point,
another series of numbers, and a null
terminator are present.
 *
 * ARGUMENTS:   char
        *ptr    pointer to string to
check.
 *
 * RETURNS:             BOOL    FALSE   if
string is not a valid non-negative
decimal value
 *
        TRUE    if string is OK
 */

BOOL IsDecimal(char *ptr)
{
        char *dotptr;
        BOOL  bValid;

        if ( *ptr == 0 )
                return FALSE;

        // find decimal point
        dotptr = strchr( ptr, '.' );
        if (dotptr == NULL)
                // no decimal point, so
just check for numeric
                return IsNumeric(ptr);
        *dotptr = 0;  // temporarily
replace decimal with a terminator

        if ( *ptr != 0 )
                bValid = IsNumeric(ptr);
        // string starts with decimal
point
        else if (*(dotptr+1) == 0)
                return FALSE;  // nothing
but a decimal point is bad
        else
                bValid = TRUE;

        if (*(dotptr+1) != 0)
                // check text after
decimal point
                bValid &=
IsNumeric(dotptr+1);

        *dotptr = '.';  // replace
decimal point
        return bValid;
}
```

Tpcc.h

```
/*      FILE:           TPCC.H
 *                              Microsoft
TPC-C Kit Ver. 4.20.000
 *                              Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                              Version
4.10.000 audited by Richard Gimarc,
Performance Metrics, 3/17/99
```

Appendix A:
Source Code

```
 *
 *      PURPOSE:       Header file for
ISAPI TPCC.DLL, defines structures and
functions used in the isapi tpcc.dll.
 *
 */

//VERSION RESOURCE DEFINES
#define _APS_NEXT_RESOURCE_VALUE
                      101
#define _APS_NEXT_COMMAND_VALUE
                              40001
#define _APS_NEXT_CONTROL_VALUE
                              1000
#define _APS_NEXT_SYMED_VALUE
                      101

#define TP_MAX_RETRIES
                              50


//note that the welcome form must be
processed first as terminal ids assigned
here, once the
//terminal id is assigned then the forms
can be processed in any order.
#define WELCOME_FORM
                      1
        //beginning form no term id
assigned, form id
#define MAIN_MENU_FORM
                      2
        //term id assigned main menu form
id
#define NEW_ORDER_FORM
                      3
        //new order form id
#define PAYMENT_FORM
                      4
        //payment form id
#define DELIVERY_FORM
                      5
        //delivery form id
#define ORDER_STATUS_FORM
                      6
        //order status id
#define STOCK_LEVEL_FORM
                      7
        //stock level form id

//This macro is used to prevent the
compiler error unused formal parameter
#define UNUSEDPARAM(x) (x = x)


//This structure defines the data
necessary to keep distinct for each
terminal or client connection.
typedef struct _CLIENTDATA
{
        int                   iNextFree;
                              //index of
next free element or -1 if this entry in
use.
        int                   w_id;
                              //warehouse
id assigned at welcome form
        int                   d_id;
                              //district
id assigned at welcome form
        int                   iSyncId;

        //syncronization id
        int                   iTickCount;
                              //time of
last access;

        CTPCC_BASE     *pTxn;

} CLIENTDATA, *PCLIENTDATA;


//This structure is used to define the
operational interface for terminal id
support
typedef struct _TERM
{
        int
        iNumEntries;
                //total allocated terminal
array entries
        int                   iFreeList;

        //next available terminal array
element or -1 if none
        int
        iMasterSyncId;
                //syncronization id
        CLIENTDATA     *pClientData;
                              //pointer
to allocated client data
} TERM;

typedef TERM *PTERM;
                              //pointer
to terminal structure type


        enum WEBERROR
        {
                NO_ERR,
                ERR_COMMAND_UNDEFINED,
                ERR_D_ID_INVALID,

        ERR_DELIVERY_CARRIER_ID_RANGE,

        ERR_DELIVERY_CARRIER_INVALID,

        ERR_DELIVERY_MISSING_OCD_KEY,

        ERR_DELIVERY_THREAD_FAILED,
                ERR_GETPROCADDR_FAILED,
                ERR_HTML_ILL_FORMED,

        ERR_INVALID_SYNC_CONNECTION,
                ERR_INVALID_TERMID,
                ERR_LOADDLL_FAILED,

        ERR_MAX_CONNECTIONS_EXCEEDED,
                ERR_MEM_ALLOC_FAILED,

        ERR_MISSING_REGISTRY_ENTRIES,

        ERR_NEWORDER_CUSTOMER_INVALID,
                ERR_NEWORDER_CUSTOMER_KEY,

        ERR_NEWORDER_DISTRICT_INVALID,

        ERR_NEWORDER_FORM_MISSING_DID,
```

```
	ERR_NEWORDER_ITEMID_INVALID,
		ERR_NEWORDER_ITEMID_RANGE,

	ERR_NEWORDER_ITEMID_WITHOUT_SUPPW
,

	ERR_NEWORDER_MISSING_IID_KEY,

	ERR_NEWORDER_MISSING_QTY_KEY,

	ERR_NEWORDER_MISSING_SUPPW_KEY,

	ERR_NEWORDER_NOITEMS_ENTERED,
		ERR_NEWORDER_QTY_INVALID,
		ERR_NEWORDER_QTY_RANGE,

	ERR_NEWORDER_QTY_WITHOUT_SUPPW,

	ERR_NEWORDER_SUPPW_INVALID,
		ERR_NO_SERVER_SPECIFIED,

	ERR_ORDERSTATUS_CID_AND_CLT,

	ERR_ORDERSTATUS_CID_INVALID,
		ERR_ORDERSTATUS_CLT_RANGE,

	ERR_ORDERSTATUS_DID_INVALID,

	ERR_ORDERSTATUS_MISSING_CID_CLT,

	ERR_ORDERSTATUS_MISSING_CID_KEY,

	ERR_ORDERSTATUS_MISSING_CLT_KEY,

	ERR_ORDERSTATUS_MISSING_DID_KEY,
		ERR_PAYMENT_CDI_INVALID,
		ERR_PAYMENT_CID_AND_CLT,

	ERR_PAYMENT_CUSTOMER_INVALID,
		ERR_PAYMENT_CWI_INVALID,

	ERR_PAYMENT_DISTRICT_INVALID,
		ERR_PAYMENT_HAM_INVALID,
		ERR_PAYMENT_HAM_RANGE,

	ERR_PAYMENT_LAST_NAME_TO_LONG,

	ERR_PAYMENT_MISSING_CDI_KEY,

	ERR_PAYMENT_MISSING_CID_CLT,

	ERR_PAYMENT_MISSING_CID_KEY,
		ERR_PAYMENT_MISSING_CLT,

	ERR_PAYMENT_MISSING_CLT_KEY,

	ERR_PAYMENT_MISSING_CWI_KEY,

	ERR_PAYMENT_MISSING_DID_KEY,

	ERR_PAYMENT_MISSING_HAM_KEY,

	ERR_STOCKLEVEL_MISSING_THRESHOLD_
KEY,

	ERR_STOCKLEVEL_THRESHOLD_INVALID,

	ERR_STOCKLEVEL_THRESHOLD_RANGE,
		ERR_VERSION_MISMATCH,
		ERR_W_ID_INVALID
	};

class CWEBCLNT_ERR : public CBaseErr
{
	public:

		CWEBCLNT_ERR(WEBERROR Err)
		{
			m_Error = Err;
			m_szTextDetail =
NULL;
			m_SystemErr = 0;
			m_szErrorText =
NULL;
		};

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

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

		WEBERROR	m_Error;
		char
	*m_szTextDetail;	//
		char
	*m_szErrorText;
		DWORD
	m_SystemErr;

		int ErrorType() {return
ERR_TYPE_WEBDLL;};
		int ErrorNum() {return
m_Error;};
		char *ErrorText();

};

//These constants have already been
defined in engstut.h, but since we do
//not want to include it in the delisrv
executable
#define	TXN_EVENT_START		2
#define	TXN_EVENT_STOP		4
#define TXN_EVENT_WARNING	6
	//used to record a warning into
the log

//function prototypes
```

Appendix A:
Source Code

```
BOOL APIENTRY DllMain(HANDLE hModule,
DWORD ul_reason_for_call, LPVOID
lpReserved);
void WriteMessageToEventLog(LPTSTR
lpszMsg);
void
ProcessQueryString(EXTENSION_CONTROL_BLOC
K *pECB, int *pCmd, int *pFormId, int
*pTermId, int *pSyncId);
void WelcomeForm(EXTENSION_CONTROL_BLOCK
*pECB, char *szBuffer);
void SubmitCmd(EXTENSION_CONTROL_BLOCK
*pECB, char *szBuffer);
void BeginCmd(EXTENSION_CONTROL_BLOCK
*pECB, int iFormId, int iTermId);
void ProcessCmd(EXTENSION_CONTROL_BLOCK
*pECB, int iFormId, int iTermId);
void StatsCmd(EXTENSION_CONTROL_BLOCK
*pECB, char *szBuffer);
void ErrorMessage(EXTENSION_CONTROL_BLOCK
*pECB, int iError, int iErrorType, char
*szMsg, int iTermId);
void GetKeyValue(char **pQueryString,
char *pKey, char *pValue, int iMax,
WEBERROR err);
int  GetIntKeyValue(char **pQueryString,
char *pKey, WEBERROR NoKeyErr, WEBERROR
NotIntErr);
void TermInit(void);
void TermDeleteAll(void);
int  TermAdd(void);
void TermDelete(int id);
void ErrorForm(EXTENSION_CONTROL_BLOCK
*pECB, int iType, int iErrorNum, int
iTermId, int iSyncId, char *szErrorText,
char *szBuffer );
void MakeMainMenuForm(int iTermId, int
iSyncId, char *szForm);
void MakeStockLevelForm(int iTermId,
STOCK_LEVEL_DATA *pStockLevelData, BOOL
bInput, char *szForm);
void MakeNewOrderForm(int iTermId,
NEW_ORDER_DATA *pNewOrderData, BOOL
bInput, char *szForm);
void MakePaymentForm(int iTermId,
PAYMENT_DATA *pPaymentData, BOOL bInput,
char *szForm);
void MakeOrderStatusForm(int iTermId,
ORDER_STATUS_DATA *pOrderStatusData, BOOL
bInput, char *szForm);
void MakeDeliveryForm(int iTermId,
DELIVERY_DATA *pDeliveryData, BOOL
bInput, char *szForm);
void
ProcessNewOrderForm(EXTENSION_CONTROL_BLO
CK     *pECB, int iTermId, char
*szBuffer);
void
ProcessPaymentForm(EXTENSION_CONTROL_BLOC
K *pECB, int iTermId, char *szBuffer);
void
ProcessOrderStatusForm(EXTENSION_CONTROL_
BLOCK *pECB, int iTermId, char
*szBuffer);
void
ProcessDeliveryForm(EXTENSION_CONTROL_BLO
CK *pECB, int iTermId, char *szBuffer);
void
ProcessStockLevelForm(EXTENSION_CONTROL_B
LOCK *pECB, int iTermId, char *szBuffer);
void GetNewOrderData(LPSTR
lpszQueryString, NEW_ORDER_DATA
*pNewOrderData);
void GetPaymentData(LPSTR
lpszQueryString, PAYMENT_DATA
*pPaymentData);
void GetOrderStatusData(LPSTR
lpszQueryString, ORDER_STATUS_DATA
*pOrderStatusData);
BOOL PostDeliveryInfo(short w_id, short
o_carrier_id);
BOOL IsNumeric(char *ptr);
BOOL IsDecimal(char *ptr);
void DeliveryWorkerThread(void *ptr);
```

Tpcc_com.cpp

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

// needed for CoinitializeEx
#define _WIN32_WINNT 0x0400

#include <windows.h>

// need to declare functions for export
#define DllDecl  __declspec( dllexport )

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

#include
"..\..\tpcc_com_ps\src\tpcc_com_ps_i.c"
#include
"..\..\tpcc_com_all\src\tpcc_com_all_i.c"

// wrapper routine for class constructor
__declspec(dllexport) CTPCC_COM*
CTPCC_COM_new(BOOL bSinglePool)
{
        return new
CTPCC_COM(bSinglePool);
}
```

Appendix A:
Source Code

```
CTPCC_COM::CTPCC_COM(BOOL bSinglePool)
{
	HRESULT hr = NULL;
	long lRet = 0;
	ULONG ulTmpSize = 0;

	m_pTxn			= NULL;
	m_pNewOrder		= NULL;
	m_pPayment		= NULL;
	m_pStockLevel  = NULL;
	m_pOrderStatus = NULL;

	m_bSinglePool  = bSinglePool;

	ulTmpSize = (ULONG)
sizeof(COM_DATA);
	VariantInit(&m_vTxn);
	m_vTxn.vt = VT_SAFEARRAY;

	m_vTxn.parray =
SafeArrayCreateVector(VT_UI1, ulTmpSize,
ulTmpSize);
	if (!m_vTxn.parray)
		throw new CCOMERR( E_FAIL
);

	memset((void*)m_vTxn.parray-
>pvData,0,ulTmpSize);
	m_pTxn =
(COM_DATA*)m_vTxn.parray->pvData;

	hr = CoInitializeEx(NULL,
COINIT_MULTITHREADED);
	if (FAILED(hr))
	{
		throw new CCOMERR( hr );
	}

	// create components
	if (m_bSinglePool)
	{
		hr =
CoCreateInstance(CLSID_TPCC, NULL,
CLSCTX_SERVER, IID_ITPCC, (void
**)&m_pNewOrder);
		if (FAILED(hr))
			throw new
CCOMERR(hr);

		// all txns will use same
component
		m_pPayment = m_pNewOrder;
		m_pStockLevel =
m_pNewOrder;
		m_pOrderStatus =
m_pNewOrder;
	}
	else
	{
		// use different
components for each txn

		hr =
CoCreateInstance(CLSID_NewOrder, NULL,
CLSCTX_SERVER, IID_ITPCC, (void
**)&m_pNewOrder);
		if (FAILED(hr))
			throw new
CCOMERR(hr);
		hr =
CoCreateInstance(CLSID_Payment, NULL,
CLSCTX_SERVER, IID_ITPCC, (void
**)&m_pPayment);
		if (FAILED(hr))
			throw new
CCOMERR(hr);

		hr =
CoCreateInstance(CLSID_StockLevel, NULL,
CLSCTX_SERVER, IID_ITPCC, (void
**)&m_pStockLevel);
		if (FAILED(hr))
			throw new
CCOMERR(hr);

		hr =
CoCreateInstance(CLSID_OrderStatus, NULL,
CLSCTX_SERVER, IID_ITPCC, (void
**)&m_pOrderStatus);
		if (FAILED(hr))
			throw new
CCOMERR(hr);
	}

	// call setcomplete to release
each component back into pool
	hr = m_pNewOrder-
>CallSetComplete();
	if (FAILED(hr))
		throw new CCOMERR(hr);

	if (!m_bSinglePool)

	{
		hr = m_pPayment-
>CallSetComplete();
		if (FAILED(hr))
			throw new
CCOMERR(hr);

		hr = m_pStockLevel-
>CallSetComplete();
		if (FAILED(hr))
			throw new
CCOMERR(hr);

		hr = m_pOrderStatus-
>CallSetComplete();
		if (FAILED(hr))
			throw new
CCOMERR(hr);
	}
}

CTPCC_COM::~CTPCC_COM()
{
	if (m_pTxn)

	SafeArrayDestroy(m_vTxn.parray);

	ReleaseInterface(m_pNewOrder);
	if (!m_bSinglePool)
	{

	ReleaseInterface(m_pPayment);

	ReleaseInterface(m_pStockLevel);

	ReleaseInterface(m_pOrderStatus);
```

```
	}
	CoUninitialize();
}

void CTPCC_COM::NewOrder()
{
	VARIANT   vTxn_out;

	HRESULT hr = m_pNewOrder-
>NewOrder(m_vTxn, &vTxn_out);
	if (FAILED(hr))
		throw new CCOMERR( hr );
	memcpy(m_pTxn, (void
*)vTxn_out.parray-
>pvData,vTxn_out.parray-
>rgsabound[0].cElements);
	SafeArrayDestroy(vTxn_out.parray)
;

	if ( m_pTxn->ErrorType !=
ERR_SUCCESS )
		throw new CCOMERR( m_pTxn-
>ErrorType, m_pTxn->error );
}

void CTPCC_COM::Payment()
{
	VARIANT   vTxn_out;

	HRESULT hr = m_pPayment-
>Payment(m_vTxn, &vTxn_out);
	if (FAILED(hr))
		throw new CCOMERR( hr );
	memcpy(m_pTxn, (void
*)vTxn_out.parray-
>pvData,vTxn_out.parray-
>rgsabound[0].cElements);
	SafeArrayDestroy(vTxn_out.parray)
;

	if ( m_pTxn->ErrorType !=
ERR_SUCCESS )
		throw new CCOMERR( m_pTxn-
>ErrorType, m_pTxn->error );
}

void CTPCC_COM::StockLevel()
{
	VARIANT   vTxn_out;

	HRESULT hr = m_pStockLevel-
>StockLevel(m_vTxn, &vTxn_out);
	if (FAILED(hr))
		throw new CCOMERR( hr );
	memcpy(m_pTxn, (void
*)vTxn_out.parray-
>pvData,vTxn_out.parray-
>rgsabound[0].cElements);
	SafeArrayDestroy(vTxn_out.parray)
;

	if ( m_pTxn->ErrorType !=
ERR_SUCCESS )
		throw new CCOMERR( m_pTxn-
>ErrorType, m_pTxn->error );
}

void CTPCC_COM::OrderStatus()
{
	VARIANT   vTxn_out;

	HRESULT hr = m_pOrderStatus-
>OrderStatus(m_vTxn, &vTxn_out);
	if (FAILED(hr))
		throw new CCOMERR( hr );
	memcpy(m_pTxn, (void
*)vTxn_out.parray-
>pvData,vTxn_out.parray-
>rgsabound[0].cElements);
	SafeArrayDestroy(vTxn_out.parray)
;

	if ( m_pTxn->ErrorType !=
ERR_SUCCESS )
		throw new CCOMERR( m_pTxn-
>ErrorType, m_pTxn->error );
}
```

Tpcc_com.h

```
/*	FILE:		TPCC_COM.H
 *				Microsoft
TPC-C Kit Ver. 4.20.000
 *				Copyright
Microsoft, 1999
 *		All Rights Reserved
 *
 *				not yet
audited
 *
 *	PURPOSE:	Header file for
TPC-C COM+ class implementation.
 *
 *  Change history:
 *		4.20.000 - first version
 */

#pragma once

#include <stdio.h>
#include
"..\..\tpcc_com_ps\src\tpcc_com_ps.h"

// need to declare functions for import,
unless define has already been created
// by the DLL's .cpp module for export.
#ifndef DllDecl

#define DllDecl  __declspec( dllimport )
#endif

class CCOMERR : public CBaseErr
{
	private:
		char m_szErrorText[64];

	public:
		// use this interface for
genuine COM errors
		CCOMERR( HRESULT hr )
		{
			m_hr = hr;
			m_iErrorType = 0;
			m_iError = 0;
		}
```

```
		// use this interface to
impersonate a non-COM error type
		CCOMERR( int iErrorType,
int iError )
		{
			m_iErrorType =
iErrorType;
			m_iError = iError;
			m_hr = S_OK;
		}

		int		m_hr;
		int
	m_iErrorType;
		int		m_iError;

		// A CCOMERR class can
impersonate another class, which happens
if the error
		// was not actually a COM
Services error, but was simply
transmitted back via COM.
		int ErrorType()
		{
			if (m_iErrorType
== 0)
				return
ERR_TYPE_COM;
			else
				return
m_iErrorType;
		}

		int ErrorNum() {return
m_hr;}

		char *ErrorText()
		{
			if (m_hr == S_OK)
				sprintf(
m_szErrorText, "Error: Class %d, error #
%d", m_iErrorType, m_iError );
			else
				sprintf(
m_szErrorText, "Error: COM HRESULT %x",
m_hr );
			return
m_szErrorText;
		}
};

class DllDecl CTPCC_COM : public
CTPCC_BASE
{
	private:
		BOOL m_bSinglePool;

		// COM Interface pointers
		ITPCC*
	m_pNewOrder;
		ITPCC*
	m_pPayment;
		ITPCC*
	m_pStockLevel;
		ITPCC*
	m_pOrderStatus;

		struct COM_DATA
		{
			int ErrorType;
			int error;
			union
			{

	NEW_ORDER_DATA		NewOrder;

	PAYMENT_DATA		Payment;

	DELIVERY_DATA		Delivery;

	STOCK_LEVEL_DATA	StockLevel;

	ORDER_STATUS_DATA
	OrderStatus;
			} u;
		} *m_pTxn;

		VARIANT m_vTxn;
	public:
		CTPCC_COM(BOOL
bSinglePool);
		~CTPCC_COM(void);

		inline PNEW_ORDER_DATA
	BuffAddr_NewOrder()		{
return &m_pTxn->u.NewOrder;	};
		inline PPAYMENT_DATA
	BuffAddr_Payment()		{
return &m_pTxn->u.Payment;	};
		inline PDELIVERY_DATA
	BuffAddr_Delivery()		{
return &m_pTxn->u.Delivery;	};
		inline PSTOCK_LEVEL_DATA
	BuffAddr_StockLevel()	{ return
&m_pTxn->u.StockLevel;	};
		inline PORDER_STATUS_DATA
	BuffAddr_OrderStatus()	{ return
&m_pTxn->u.OrderStatus; };

		void NewOrder		();
		void Payment		();
		void StockLevel
	();
		void OrderStatus	();
		void Delivery		()
{ throw new CCOMERR(E_NOTIMPL); }  // not
supported
};


inline void ReleaseInterface(IUnknown
*pUnk)
{
	if (pUnk)

	{
		pUnk->Release();
		pUnk = NULL;
	}
}


// wrapper routine for class constructor
extern "C" __declspec(dllexport)
CTPCC_COM* CTPCC_COM_new(BOOL);

typedef CTPCC_COM*
(TYPE_CTPCC_COM)(BOOL);
```

Appendix A:
Source Code

Tpcc_com_all.cpp

```
/*      FILE:           TPCC_COM_ALL.CPP
 *                              Microsoft
TPC-C Kit Ver. 4.20.000
 *                              Copyright
Microsoft, 1999
 *              All Rights Reserved
 *
 *                              Version
4.10.000 audited by Richard Gimarc,
Performance Metrics, 3/17/99
 *
 *      PURPOSE:        Implementation for
TPC-C Tuxedo class.
 *      Contact:        Charles Levine
(clevine@microsoft.com)
 *
 *  Change history:
 *              4.20.000 - updated rev
number to match kit
 */

#define STRICT
#define _WIN32_WINNT 0x0400
#define _ATL_APARTMENT_THREADED

#include <stdio.h>
#include <atlbase.h>
//You may derive a class from CComModule
and use it if you want to override
//something, but do not change the name
of _Module
extern CComModule _Module;

#include <atlcom.h>
#include <initguid.h>
#include <transact.h>
#include <atlimpl.cpp>
#include <comsvcs.h>

#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>

#include "tpcc_com_ps.h"
#include "..\..\common\src\trans.h"
                                //tpckit
transaction header contains definations
of structures specific to TPC-C
#include "..\..\common\src\txn_base.h"
#include "..\..\common\src\error.h"
#include
"..\..\common\src\ReadRegistry.h"
#include
"..\..\db_dblib_dll\src\tpcc_dblib.h"
        // DBLIB implementation of TPC-C
txns
#include
"..\..\db_odbc_dll\src\tpcc_odbc.h"
        // ODBC implementation of TPC-C
txns

#include "resource.h"
#include "tpcc_com_all.h"
#include "tpcc_com_all_i.c"
#include "Methods.h"
#include
"..\..\tpcc_com_ps\src\tpcc_com_ps_i.c"
#include
"..\..\common\src\ReadRegistry.cpp"


CComModule _Module;


BEGIN_OBJECT_MAP(ObjectMap)
        OBJECT_ENTRY(CLSID_TPCC, CTPCC)
        OBJECT_ENTRY(CLSID_NewOrder,
CNewOrder)
        OBJECT_ENTRY(CLSID_OrderStatus,
COrderStatus)
        OBJECT_ENTRY(CLSID_Payment,
CPayment)
        OBJECT_ENTRY(CLSID_StockLevel,
CStockLevel)
END_OBJECT_MAP()


// configuration settings from registry
TPCCREGISTRYDATA        Reg;
char
        szMyComputerName[MAX_COMPUTERNAME
_LENGTH+1];

static  HINSTANCE hLibInstanceDb = NULL;

TYPE_CTPCC_DBLIB        *pCTPCC_DBLIB_new;
TYPE_CTPCC_ODBC
        *pCTPCC_ODBC_new;


/////////////////////////////////////////
/////////////////////////////////////
// DLL Entry Point

extern "C"
BOOL WINAPI DllMain(HINSTANCE hInstance,
DWORD dwReason, LPVOID /*lpReserved*/)
{
        char szDllName[128];

        try
        {
                if (dwReason ==
DLL_PROCESS_ATTACH)
                {

        _Module.Init(ObjectMap,
hInstance);

        DisableThreadLibraryCalls(hInstan
ce);

                        DWORD dwSize =
MAX_COMPUTERNAME_LENGTH+1;

        GetComputerName(szMyComputerName,
&dwSize);

        szMyComputerName[dwSize] = 0;

                        if (
ReadTPCCRegistrySettings( &Reg ) )
                                throw new
CCOMPONENT_ERR(
ERR_MISSING_REGISTRY_ENTRIES );
```

```
                        if
(Reg.eDB_Protocol == DBLIB)
                        {
                                strcpy(
szDllName, Reg.szPath );
                                strcat(
szDllName, "tpcc_dblib.dll");

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

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

                                // get
function pointer to wrapper for class
constructor

        pCTPCC_DBLIB_new =
(TYPE_CTPCC_DBLIB*)
GetProcAddress(hLibInstanceDb,"CTPCC_DBLI
B_new");
                                if
(pCTPCC_DBLIB_new == NULL)

        throw new CCOMPONENT_ERR(
ERR_GETPROCADDR_FAILED, szDllName,
GetLastError() );
                        }
                        else if
(Reg.eDB_Protocol == ODBC)
                        {
                                strcpy(
szDllName, Reg.szPath );
                                strcat(
szDllName, "tpcc_odbc.dll");

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

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

                                // get
function pointer to wrapper for class
constructor

        pCTPCC_ODBC_new =
(TYPE_CTPCC_ODBC*)
GetProcAddress(hLibInstanceDb,"CTPCC_ODBC
_new");
                                if
(pCTPCC_ODBC_new == NULL)

        throw new CCOMPONENT_ERR(
ERR_GETPROCADDR_FAILED, szDllName,
GetLastError() );
                        }
                        else
                                throw new
CCOMPONENT_ERR( ERR_UNKNOWN_DB_PROTOCOL
);
                }
                else if (dwReason ==
DLL_PROCESS_DETACH)
                        _Module.Term();

        }
        catch (CBaseErr *e)
        {
                WriteMessageToEventLog(e-
>ErrorText());
                delete e;
                return FALSE;
        }
        catch (...)
        {

        WriteMessageToEventLog(TEXT("Unha
ndled exception in object DllMain"));
                return FALSE;
        }

        return TRUE;    // OK
}

/////////////////////////////////////////
////////////////////////////////////
// Used to determine whether the DLL can
be unloaded by OLE

STDAPI DllCanUnloadNow(void)
{
        return
(_Module.GetLockCount()==0) ? S_OK :
S_FALSE;
}

/////////////////////////////////////////
////////////////////////////////////
// Returns a class factory to create an
object of the requested type

STDAPI DllGetClassObject(REFCLSID rclsid,
REFIID riid, LPVOID* ppv)
{
        return
_Module.GetClassObject(rclsid, riid,
ppv);
}

/////////////////////////////////////////
////////////////////////////////////
// DllRegisterServer - Adds entries to
the system registry

STDAPI DllRegisterServer(void)
{
        // registers object, typelib and
all interfaces in typelib
        return
_Module.RegisterServer(TRUE);
}

/////////////////////////////////////////
////////////////////////////////////
// DllUnregisterServer - Removes entries
from the system registry

STDAPI DllUnregisterServer(void)
{
        _Module.UnregisterServer();
        return S_OK;
```

```
}

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

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

    _stprintf(szMsg, TEXT("Error in COM+
TPC-C Component: "));
    lpszStrings[0] = szMsg;
    lpszStrings[1] = lpszMsg;

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

        (VOID)
DeregisterEventSource(hEventSource);
    }
}

inline void ReleaseInterface(IUnknown
*pUnk)
{
        if (pUnk)
        {
                pUnk->Release();
                pUnk = NULL;
        }
}

/* FUNCTION: CCOMPONENT_ERR::ErrorText
 *
 */

char* CCOMPONENT_ERR::ErrorText(void)
{
        static SERRORMSG errorMsgs[] =
        {
                {
ERR_MISSING_REGISTRY_ENTRIES, "Required
entries missing from registry."
                        },
                { ERR_LOADDLL_FAILED,
                "Load of DLL failed.
DLL="
                        },
                { ERR_GETPROCADDR_FAILED,
                "Could not map proc in
DLL.  GetProcAddr error.  DLL="      },
                { ERR_UNKNOWN_DB_PROTOCOL,
                "Unknown database protocol
specified in registry."             },
                { 0,
                                ""


                        }
        };

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

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

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


CTPCC_Common::CTPCC_Common()
{
        m_pTxn = NULL;
        m_bCanBePooled = TRUE;
}

CTPCC_Common::~CTPCC_Common()
{
        if (m_pTxn)
                delete m_pTxn;
}

HRESULT CTPCC_Common::CallSetComplete()
{
        IObjectContext* pObjectContext =
NULL;
```

Appendix A: Source Code

```
	// get our object context
	HRESULT hr = CoGetObjectContext(
IID_IObjectContext, (void
**)&pObjectContext );
	pObjectContext->SetComplete();
	ReleaseInterface(pObjectContext);
	return hr;
}

//
//	called by the ctor activator

//
STDMETHODIMP
CTPCC_Common::Construct(IDispatch * pUnk)
{
	// Code to access construction
string, if needed later...
	//	if (!pUnk)
	//		return
E_UNEXPECTED;
	//	IObjectConstructString *
pString = NULL;
	//	HRESULT hr = pUnk-
>QueryInterface(IID_IObjectConstructStrin
g, (void **)&pString);
	//	pString->Release();

	try
	{
		if (Reg.eDB_Protocol ==
ODBC)
			m_pTxn =
pCTPCC_ODBC_new( Reg.szDbServer,
Reg.szDbUser, Reg.szDbPassword,
szMyComputerName, Reg.szDbName );
		else if (Reg.eDB_Protocol
== DBLIB)
			m_pTxn =
pCTPCC_DBLIB_new( Reg.szDbServer,
Reg.szDbUser, Reg.szDbPassword,
szMyComputerName, Reg.szDbName );
	}
	catch (CBaseErr *e)
	{
		WriteMessageToEventLog(e-
>ErrorText());
		delete e;
		return E_FAIL;
	}
	catch (...)
	{

	WriteMessageToEventLog(TEXT("Unha
ndled exception in object ::Construct"));
		return E_FAIL;
	}

	return S_OK;
}

HRESULT CTPCC_Common::NewOrder(VARIANT
txn_in, VARIANT* txn_out)
{
	PNEW_ORDER_DATA		pNewOrder;
	COM_DATA		*pData;
	try
	{
		pData =
(COM_DATA*)txn_in.parray->pvData;
		pNewOrder = m_pTxn-
>BuffAddr_NewOrder();

		memcpy(pNewOrder, &pData-
>u.NewOrder, sizeof(NEW_ORDER_DATA));

		m_pTxn->NewOrder();
	// do the actual txn

		VariantInit(txn_out);
		txn_out->vt =
VT_SAFEARRAY;
		txn_out->parray =
SafeArrayCreateVector(VT_UI1,

		txn_in.parray->rgsabound-
>cElements,

		txn_in.parray->rgsabound-
>cElements);
		pData = (COM_DATA*)
txn_out->parray->pvData;

		memcpy( &pData-
>u.NewOrder, pNewOrder,
sizeof(NEW_ORDER_DATA));

		pData->retval =
ERR_SUCCESS;
		pData->error = 0;
		return S_OK;
	}
	catch (CBaseErr *e)
	{
		// check for lost database
connection; if yes, component is toast
		if ( ((e->ErrorType() ==
ERR_TYPE_DBLIB) && (e->ErrorNum() ==
10005)) ||
			((e->ErrorType()
== ERR_TYPE_ODBC) && (e->ErrorNum() ==
10054)) )
			m_bCanBePooled =
FALSE;

		pData->retval = e-
>ErrorType();
		pData->error = e-
>ErrorNum();
		delete e;
		return E_FAIL;
	}
	catch (...)
	{

	WriteMessageToEventLog(TEXT("Unha
ndled exception."));
		pData->retval =
ERR_TYPE_LOGIC;
		pData->error = 0;
		m_bCanBePooled = FALSE;
		return E_FAIL;
	}
}

HRESULT CTPCC_Common::Payment(VARIANT
txn_in, VARIANT* txn_out)
{
	PPAYMENT_DATA	pPayment;
	COM_DATA		*pData;
```

```
        try
        {
                pData =
(COM_DATA*)txn_in.parray->pvData;
                pPayment = m_pTxn-
>BuffAddr_Payment();

                memcpy(pPayment, &pData-
>u.Payment, sizeof(PAYMENT_DATA));

                m_pTxn->Payment();
        // do the actual txn

                VariantInit(txn_out);
                txn_out->vt =
VT_SAFEARRAY;
                txn_out->parray =
SafeArrayCreateVector( VT_UI1,

                txn_in.parray->rgsabound-
>cElements,

                txn_in.parray->rgsabound-
>cElements);
                pData = (COM_DATA*)
txn_out->parray->pvData;

                memcpy( &pData->u.Payment,
pPayment, sizeof(PAYMENT_DATA));

                pData->retval =
ERR_SUCCESS;
                pData->error = 0;
                return S_OK;
        }
        catch (CBaseErr *e)
        {
                // check for lost database
connection; if yes, component is toast
                if ( ((e->ErrorType() ==
ERR_TYPE_DBLIB) && (e->ErrorNum() ==
10005)) ||
                         ((e->ErrorType()
== ERR_TYPE_ODBC) && (e->ErrorNum() ==
10054)) )
                        m_bCanBePooled =
FALSE;

                pData->retval = e-
>ErrorType();
                pData->error = e-
>ErrorNum();
                delete e;
                return E_FAIL;
        }
        catch (...)
        {

        WriteMessageToEventLog(TEXT("Unha
ndled exception."));
                pData->retval =
ERR_TYPE_LOGIC;
                pData->error = 0;
                m_bCanBePooled = FALSE;
                return E_FAIL;
        }
}

HRESULT CTPCC_Common::StockLevel(VARIANT
txn_in, VARIANT* txn_out)
{
        PSTOCK_LEVEL_DATA
        pStockLevel;
        COM_DATA
        *pData;

        try
        {
                pData =
(COM_DATA*)txn_in.parray->pvData;
                pStockLevel = m_pTxn-
>BuffAddr_StockLevel();

                memcpy(pStockLevel,
&pData->u.StockLevel,
sizeof(STOCK_LEVEL_DATA));

                m_pTxn->StockLevel();

                VariantInit(txn_out);
                txn_out->vt =
VT_SAFEARRAY;
                txn_out->parray =
SafeArrayCreateVector( VT_UI1,

                txn_in.parray->rgsabound-
>cElements,

                txn_in.parray->rgsabound-
>cElements);
                pData =
(COM_DATA*)txn_out->parray->pvData;

                memcpy( &pData-
>u.StockLevel, pStockLevel,
sizeof(STOCK_LEVEL_DATA));

                pData->retval =
ERR_SUCCESS;
                pData->error = 0;
                return S_OK;
        }
        catch (CBaseErr *e)
        {
                // check for lost database
connection; if yes, component is toast
                if ( ((e->ErrorType() ==
ERR_TYPE_DBLIB) && (e->ErrorNum() ==
10005)) ||
                         ((e->ErrorType()
== ERR_TYPE_ODBC) && (e->ErrorNum() ==
10054)) )
                        m_bCanBePooled =
FALSE;

                pData->retval = e-
>ErrorType();
                pData->error = e-
>ErrorNum();
                delete e;
                return E_FAIL;
        }
        catch (...)
        {

        WriteMessageToEventLog(TEXT("Unha
ndled exception."));
                pData->retval =
ERR_TYPE_LOGIC;
                pData->error = 0;
```

```
                m_bCanBePooled = FALSE;
                return E_FAIL;
        }
}

HRESULT CTPCC_Common::OrderStatus(VARIANT
txn_in, VARIANT* txn_out)
{
        PORDER_STATUS_DATA
        pOrderStatus;
        COM_DATA
        *pData;
        try
        {
                pData =
(COM_DATA*)txn_in.parray->pvData;
                pOrderStatus = m_pTxn-
>BuffAddr_OrderStatus();

                memcpy(pOrderStatus,
&pData->u.OrderStatus,
sizeof(ORDER_STATUS_DATA));

                m_pTxn->OrderStatus();

                VariantInit(txn_out);
                txn_out->vt =
VT_SAFEARRAY;
                txn_out->parray =
SafeArrayCreateVector( VT_UI1,

                txn_in.parray->rgsabound-
>cElements,

                txn_in.parray->rgsabound-
>cElements);
                pData =
(COM_DATA*)txn_out->parray->pvData;

                memcpy( &pData-
>u.OrderStatus, pOrderStatus,
sizeof(ORDER_STATUS_DATA));

                pData->retval =
ERR_SUCCESS;
                pData->error = 0;
                return S_OK;
        }
        catch (CBaseErr *e)
        {
                // check for lost database
connection; if yes, component is toast
                if ( ((e->ErrorType() ==
ERR_TYPE_DBLIB) && (e->ErrorNum() ==
10005)) ||
                         ((e->ErrorType()
== ERR_TYPE_ODBC) && (e->ErrorNum() ==
10054)) )
                        m_bCanBePooled =
FALSE;

                pData->retval = e-
>ErrorType();
                pData->error = e-
>ErrorNum();
                delete e;
                return E_FAIL;
        }
        catch (...)
        {

        WriteMessageToEventLog(TEXT("Unha
ndled exception."));
                pData->retval =
ERR_TYPE_LOGIC;
                pData->error = 0;
                m_bCanBePooled = FALSE;
                return E_FAIL;
        }
}
```

Tpcc_com_all.h

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

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


 /* File created by MIDL compiler version
5.03.0280 */
/* at Thu Dec 13 23:13:14 2001
 */
/* Compiler settings for
.\src\tpcc_com_all.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32
(32b run), ms_ext, c_ext
    error checks: allocation ref
bounds_check enum stub_data
    VC __declspec() decoration level:
         __declspec(uuid()),
__declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(),
MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )


/* verify that the <rpcndr.h> version is
high enough to compile this file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 440
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __tpcc_com_all_h__
#define __tpcc_com_all_h__

/* Forward Declarations */

#ifndef __TPCC_FWD_DEFINED__
#define __TPCC_FWD_DEFINED__

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

#endif  /* __TPCC_FWD_DEFINED__ */


#ifndef __NewOrder_FWD_DEFINED__
```

```
#define __NewOrder_FWD_DEFINED__

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

#endif  /* __NewOrder_FWD_DEFINED__ */


#ifndef __OrderStatus_FWD_DEFINED__
#define __OrderStatus_FWD_DEFINED__

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

#endif  /* __OrderStatus_FWD_DEFINED__ */


#ifndef __Payment_FWD_DEFINED__
#define __Payment_FWD_DEFINED__

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

#endif  /* __Payment_FWD_DEFINED__ */


#ifndef __StockLevel_FWD_DEFINED__
#define __StockLevel_FWD_DEFINED__

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

#endif  /* __StockLevel_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"
#include "tpcc_com_ps.h"

#ifdef __cplusplus
extern "C"{
#endif

void __RPC_FAR * __RPC_USER
MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void
__RPC_FAR * );

/* interface __MIDL_itf_tpcc_com_all_0000
*/
/* [local] */

extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_all_0000_v0_0_c_ifspe
c;
extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_all_0000_v0_0_s_ifspe
c;


#ifndef __TPCCLib_LIBRARY_DEFINED__
#define __TPCCLib_LIBRARY_DEFINED__

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


EXTERN_C const IID LIBID_TPCCLib;

EXTERN_C const CLSID CLSID_TPCC;

#ifdef __cplusplus

class DECLSPEC_UUID("122A3128-2520-11D3-
BA71-00C04FBFE08B")
TPCC;
#endif

EXTERN_C const CLSID CLSID_NewOrder;

#ifdef __cplusplus

class DECLSPEC_UUID("975BAABF-84A7-11D2-
BA47-00C04FBFE08B")
NewOrder;
#endif

EXTERN_C const CLSID CLSID_OrderStatus;

#ifdef __cplusplus

class DECLSPEC_UUID("266836AD-A50D-11D2-
BA4E-00C04FBFE08B")
OrderStatus;
#endif

EXTERN_C const CLSID CLSID_Payment;

#ifdef __cplusplus

class DECLSPEC_UUID("CD02F7EF-A4FA-11D2-
BA4E-00C04FBFE08B")
Payment;
#endif

EXTERN_C const CLSID CLSID_StockLevel;

#ifdef __cplusplus

class DECLSPEC_UUID("2668369E-A50D-11D2-
BA4E-00C04FBFE08B")
StockLevel;
#endif
#endif /* __TPCCLib_LIBRARY_DEFINED__ */

/* Additional Prototypes for ALL
interfaces */

/* end of Additional Prototypes */
```

Appendix A: Source Code

```
#ifdef __cplusplus
}
#endif

#endif
```

Resource.h

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

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE
202
#define _APS_NEXT_COMMAND_VALUE
32768
#define _APS_NEXT_CONTROL_VALUE
201
#define _APS_NEXT_SYMED_VALUE
106
#endif
#endif
```

Methods.h

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


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

class CCOMPONENT_ERR : public CBaseErr
{
        public:

        CCOMPONENT_ERR(COMPONENT_ERROR
Err)
                {
                        m_Error = Err;
                        m_szTextDetail =
NULL;
                        m_SystemErr = 0;
                        m_szErrorText =
NULL;
                };


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

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

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

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


static void WriteMessageToEventLog(LPTSTR
lpszMsg);


/////////////////////////////////////////
//////////////////////////////////
// CTPCC_Common
class CTPCC_Common :
```

```
	public ITPCC,
	public IObjectControl,
	public IObjectConstruct,
	public
CComObjectRootEx<CComSingleThreadModel>
{
public:
BEGIN_COM_MAP(CTPCC_Common)
	COM_INTERFACE_ENTRY(ITPCC)
	COM_INTERFACE_ENTRY(IObjectContro
l)
	COM_INTERFACE_ENTRY(IObjectConstr
uct)
END_COM_MAP()

	CTPCC_Common();
	~CTPCC_Common();

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

	HRESULT __stdcall
CallSetComplete();

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

// IObjectConstruct
	STDMETHODIMP Construct(IDispatch
* pUnk);

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


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

};

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

BEGIN_COM_MAP(CTPCC)
	COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
	COM_INTERFACE_ENTRY_CHAIN(CTPCC_C
ommon)
END_COM_MAP()

};


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

BEGIN_COM_MAP(CNewOrder)
	COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
	COM_INTERFACE_ENTRY_CHAIN(CTPCC_C
ommon)
END_COM_MAP()

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


	/////////////////////////////////
/////////////////////////////////////////
///
// COrderStatus
class COrderStatus :
```

```
	public CTPCC_Common,
	public CComCoClass<COrderStatus,
&CLSID_OrderStatus>
{
public:
DECLARE_REGISTRY_RESOURCEID(IDR_ORDERSTAT
US)

BEGIN_COM_MAP(COrderStatus)
	COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
	COM_INTERFACE_ENTRY_CHAIN(CTPCC_C
ommon)
END_COM_MAP()

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


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

BEGIN_COM_MAP(CPayment)
	COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
	COM_INTERFACE_ENTRY_CHAIN(CTPCC_C
ommon)
END_COM_MAP()

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


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

BEGIN_COM_MAP(CStockLevel)
	COM_INTERFACE_ENTRY2(IUnknown,
CComObjectRootEx)
	COM_INTERFACE_ENTRY_CHAIN(CTPCC_C
ommon)
END_COM_MAP()

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

Tpcc_com_i.c

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

/* 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
5.03.0280 */
/* at Thu Dec 13 23:13:14 2001
 */
/* Compiler settings for
.\src\tpcc_com_all.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32
(32b run), ms_ext, c_ext
    error checks: allocation ref
bounds_check enum stub_data
    VC __declspec() decoration level:
         __declspec(uuid()),
__declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(),
MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if !defined(_M_IA64) &&
!defined(_M_AXP64)

#ifdef __cplusplus
```

Appendix A: Source Code

```
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,b
6,b7,b8)

#else // !_MIDL_USE_GUIDDEF_

#ifndef __IID_DEFINED__
#define __IID_DEFINED__

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

#endif // __IID_DEFINED__

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

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

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
LIBID_TPCCLib,0x122A3117,0x2520,0x11D3,0x
BA,0x71,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_TPCC,0x122A3128,0x2520,0x11D3,0xBA,
0x71,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_NewOrder,0x975BAABF,0x84A7,0x11D2,0
xBA,0x47,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_OrderStatus,0x266836AD,0xA50D,0x11D
2,0xBA,0x4E,0x00,0xC0,0x4F,0xBF,0xE0,0x8B
);


MIDL_DEFINE_GUID(CLSID,
CLSID_Payment,0xCD02F7EF,0xA4FA,0x11D2,0x
BA,0x4E,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_StockLevel,0x2668369E,0xA50D,0x11D2
,0xBA,0x4E,0x00,0xC0,0x4F,0xBF,0xE0,0x8B)
;

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif



#endif /* !defined(_M_IA64) &&
!defined(_M_AXP64)*/


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

/* 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
5.03.0280 */
/* at Thu Dec 13 23:13:14 2001
 */
/* Compiler settings for
.\src\tpcc_com_all.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win64
(32b run,appending), 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_AXP64)

#ifdef __cplusplus
extern "C"{
#endif


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

#ifdef _MIDL_USE_GUIDDEF_

#ifndef INITGUID
#define INITGUID
```

Appendix A: Source Code

```
#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,b
6,b7,b8)

#else // !_MIDL_USE_GUIDDEF_

#ifndef __IID_DEFINED__
#define __IID_DEFINED__

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

#endif // __IID_DEFINED__

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

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

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
LIBID_TPCCLib,0x122A3117,0x2520,0x11D3,0x
BA,0x71,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_TPCC,0x122A3128,0x2520,0x11D3,0xBA,
0x71,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_NewOrder,0x975BAABF,0x84A7,0x11D2,0
xBA,0x47,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_OrderStatus,0x266836AD,0xA50D,0x11D
2,0xBA,0x4E,0x00,0xC0,0x4F,0xBF,0xE0,0x8B
);


MIDL_DEFINE_GUID(CLSID,
CLSID_Payment,0xCD02F7EF,0xA4FA,0x11D2,0x
BA,0x4E,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);


MIDL_DEFINE_GUID(CLSID,
CLSID_StockLevel,0x2668369E,0xA50D,0x11D2
,0xBA,0x4E,0x00,0xC0,0x4F,0xBF,0xE0,0x8B)
;

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif



#endif /* defined(_M_IA64) ||
defined(_M_AXP64)*/
```

Tpcc_com_ps.h

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

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


 /* File created by MIDL compiler version
5.03.0280 */
/* at Thu Dec 13 23:13:08 2001
 */
/* Compiler settings for
.\src\tpcc_com_ps.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32
(32b run), ms_ext, c_ext
    error checks: allocation ref
bounds_check enum stub_data
    VC __declspec() decoration level:
         __declspec(uuid()),
__declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(),
MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )


/* verify that the <rpcndr.h> version is
high enough to compile this file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 440
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __RPCNDR_H_VERSION__
#error this stub requires an updated
version of <rpcndr.h>
#endif // __RPCNDR_H_VERSION__

#ifndef COM_NO_WINDOWS_H
#include "windows.h"
#include "ole2.h"
#endif /*COM_NO_WINDOWS_H*/

#ifndef __tpcc_com_ps_h__
#define __tpcc_com_ps_h__

/* Forward Declarations */
```

Appendix A: Source Code

```
#ifndef __ITPCC_FWD_DEFINED__
#define __ITPCC_FWD_DEFINED__
typedef interface ITPCC ITPCC;
#endif /* __ITPCC_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"

#ifdef __cplusplus
extern "C"{
#endif

void __RPC_FAR * __RPC_USER
MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void
__RPC_FAR * );

/* interface __MIDL_itf_tpcc_com_ps_0000
*/
/* [local] */




extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_ps_0000_v0_0_c_ifspec
;
extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_ps_0000_v0_0_s_ifspec
;

#ifndef __ITPCC_INTERFACE_DEFINED__
#define __ITPCC_INTERFACE_DEFINED__

/* interface ITPCC */
/*
[unique][helpstring][uuid][oleautomation]
[object] */


EXTERN_C const IID IID_ITPCC;

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

    MIDL_INTERFACE("FEEE6AA2-84B1-11d2-
BA47-00C04FBFE08B")
    ITPCC : public IUnknown
    {
    public:
        virtual HRESULT __stdcall
NewOrder(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
Payment(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
Delivery(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
StockLevel(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
OrderStatus(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
CallSetComplete( void) = 0;

    };

#else   /* C style interface */

    typedef struct ITPCCVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *QueryInterface )(
            ITPCC __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void
__RPC_FAR *__RPC_FAR *ppvObject);

        ULONG ( STDMETHODCALLTYPE
__RPC_FAR *AddRef )(
            ITPCC __RPC_FAR * This);

        ULONG ( STDMETHODCALLTYPE
__RPC_FAR *Release )(
            ITPCC __RPC_FAR * This);

        HRESULT ( __stdcall __RPC_FAR
*NewOrder )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*Payment )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*Delivery )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*StockLevel )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*OrderStatus )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
```

Appendix A: Source Code

```
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*CallSetComplete )(
            ITPCC __RPC_FAR * This);

        END_INTERFACE
    } ITPCCVtbl;

    interface ITPCC
    {
        CONST_VTBL struct ITPCCVtbl
__RPC_FAR *lpVtbl;
    };



#ifdef COBJMACROS


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

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

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


#define
ITPCC_NewOrder(This,txn_in,txn_out)   \
    (This)->lpVtbl ->
NewOrder(This,txn_in,txn_out)

#define
ITPCC_Payment(This,txn_in,txn_out)    \
    (This)->lpVtbl ->
Payment(This,txn_in,txn_out)

#define
ITPCC_Delivery(This,txn_in,txn_out)   \
    (This)->lpVtbl ->
Delivery(This,txn_in,txn_out)

#define
ITPCC_StockLevel(This,txn_in,txn_out) \
    (This)->lpVtbl ->
StockLevel(This,txn_in,txn_out)

#define
ITPCC_OrderStatus(This,txn_in,txn_out)
        \
    (This)->lpVtbl ->
OrderStatus(This,txn_in,txn_out)

#define ITPCC_CallSetComplete(This)   \
    (This)->lpVtbl ->
CallSetComplete(This)

#endif /* COBJMACROS */


#endif  /* C style interface */

HRESULT __stdcall ITPCC_NewOrder_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall ITPCC_Payment_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall ITPCC_Delivery_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,

    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall ITPCC_StockLevel_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall
ITPCC_OrderStatus_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT __RPC_FAR
*txn_out);
```

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


HRESULT __stdcall
ITPCC_CallSetComplete_Proxy(
    ITPCC __RPC_FAR * This);


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



#endif  /* __ITPCC_INTERFACE_DEFINED__ */


/* Additional Prototypes for ALL
interfaces */

unsigned long             __RPC_USER
VARIANT_UserSize(     unsigned long
__RPC_FAR *, unsigned long            ,
VARIANT __RPC_FAR * );
unsigned char __RPC_FAR * __RPC_USER
VARIANT_UserMarshal(  unsigned long
__RPC_FAR *, unsigned char __RPC_FAR *,
VARIANT __RPC_FAR * );
unsigned char __RPC_FAR * __RPC_USER
VARIANT_UserUnmarshal(unsigned long
__RPC_FAR *, unsigned char __RPC_FAR *,
VARIANT __RPC_FAR * );
void                      __RPC_USER
VARIANT_UserFree(     unsigned long
__RPC_FAR *, VARIANT __RPC_FAR * );

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

Dlldata.c

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

        DO NOT ALTER THIS FILE

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

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

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


#include <rpcproxy.h>

#ifdef __cplusplus
extern "C"   {
#endif


EXTERN_PROXY_FILE( tpcc_com_ps )


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


DLLDATA_ROUTINES( aProxyFileList,
GET_DLL_CLSID )

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

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

Tpcc_com_ps.h

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

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


 /* File created by MIDL compiler version
5.03.0280 */
/* at Thu Dec 13 23:13:08 2001
 */
/* Compiler settings for
.\src\tpcc_com_ps.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32
(32b run), ms_ext, c_ext
    error checks: allocation ref
bounds_check enum stub_data
    VC __declspec() decoration level:
         __declspec(uuid()),
__declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(),
MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )


/* verify that the <rpcndr.h> version is
high enough to compile this file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 440
```

Appendix A:
Source Code

```
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __RPCNDR_H_VERSION__
#error this stub requires an updated
version of <rpcndr.h>
#endif // __RPCNDR_H_VERSION__

#ifndef COM_NO_WINDOWS_H
#include "windows.h"
#include "ole2.h"
#endif /*COM_NO_WINDOWS_H*/

#ifndef __tpcc_com_ps_h__
#define __tpcc_com_ps_h__

/* Forward Declarations */

#ifndef __ITPCC_FWD_DEFINED__
#define __ITPCC_FWD_DEFINED__
typedef interface ITPCC ITPCC;
#endif  /* __ITPCC_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"

#ifdef __cplusplus
extern "C"{
#endif

void __RPC_FAR * __RPC_USER
MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void
__RPC_FAR * );

/* interface __MIDL_itf_tpcc_com_ps_0000
*/
/* [local] */




extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_ps_0000_v0_0_c_ifspec
;
extern RPC_IF_HANDLE
__MIDL_itf_tpcc_com_ps_0000_v0_0_s_ifspec
;

#ifndef __ITPCC_INTERFACE_DEFINED__
#define __ITPCC_INTERFACE_DEFINED__

/* interface ITPCC */
/*
[unique][helpstring][uuid][oleautomation]
[object] */


EXTERN_C const IID IID_ITPCC;

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

    MIDL_INTERFACE("FEEE6AA2-84B1-11d2-
BA47-00C04FBFE08B")
    ITPCC : public IUnknown
    {
    public:
        virtual HRESULT __stdcall
NewOrder(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
Payment(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
Delivery(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
StockLevel(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
OrderStatus(
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out) = 0;

        virtual HRESULT __stdcall
CallSetComplete( void) = 0;

    };

#else   /* C style interface */

    typedef struct ITPCCVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *QueryInterface )(
            ITPCC __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void
__RPC_FAR *__RPC_FAR *ppvObject);

        ULONG ( STDMETHODCALLTYPE
__RPC_FAR *AddRef )(
            ITPCC __RPC_FAR * This);

        ULONG ( STDMETHODCALLTYPE
__RPC_FAR *Release )(
            ITPCC __RPC_FAR * This);

        HRESULT ( __stdcall __RPC_FAR
*NewOrder )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*Payment )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
```

```
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*Delivery )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*StockLevel )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*OrderStatus )(
            ITPCC __RPC_FAR * This,
            /* [in] */ VARIANT txn_in,
            /* [out] */ VARIANT __RPC_FAR
*txn_out);

        HRESULT ( __stdcall __RPC_FAR
*CallSetComplete )(
            ITPCC __RPC_FAR * This);

        END_INTERFACE
    } ITPCCVtbl;

    interface ITPCC
    {
        CONST_VTBL struct ITPCCVtbl
__RPC_FAR *lpVtbl;
    };



#ifdef COBJMACROS


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

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

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


#define
ITPCC_NewOrder(This,txn_in,txn_out)   \
    (This)->lpVtbl ->
NewOrder(This,txn_in,txn_out)

#define
ITPCC_Payment(This,txn_in,txn_out)    \
    (This)->lpVtbl ->
Payment(This,txn_in,txn_out)

#define
ITPCC_Delivery(This,txn_in,txn_out)   \
    (This)->lpVtbl ->
Delivery(This,txn_in,txn_out)

#define
ITPCC_StockLevel(This,txn_in,txn_out) \
    (This)->lpVtbl ->
StockLevel(This,txn_in,txn_out)

#define
ITPCC_OrderStatus(This,txn_in,txn_out)
       \
    (This)->lpVtbl ->
OrderStatus(This,txn_in,txn_out)

#define ITPCC_CallSetComplete(This)   \
    (This)->lpVtbl ->
CallSetComplete(This)

#endif /* COBJMACROS */


#endif  /* C style interface */



HRESULT __stdcall ITPCC_NewOrder_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall ITPCC_Payment_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall ITPCC_Delivery_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,

    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall ITPCC_StockLevel_Proxy(
    ITPCC __RPC_FAR * This,
```

```
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall
ITPCC_OrderStatus_Proxy(
    ITPCC __RPC_FAR * This,
    /* [in] */ VARIANT txn_in,
    /* [out] */ VARIANT __RPC_FAR
*txn_out);


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


HRESULT __stdcall
ITPCC_CallSetComplete_Proxy(
    ITPCC __RPC_FAR * This);


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



#endif  /* __ITPCC_INTERFACE_DEFINED__ */


/* Additional Prototypes for ALL
interfaces */

unsigned long             __RPC_USER
VARIANT_UserSize(     unsigned long
__RPC_FAR *, unsigned long            ,
VARIANT __RPC_FAR * );
unsigned char __RPC_FAR * __RPC_USER
VARIANT_UserMarshal(  unsigned long
__RPC_FAR *, unsigned char __RPC_FAR *,
VARIANT __RPC_FAR * );
unsigned char __RPC_FAR * __RPC_USER
VARIANT_UserUnmarshal(unsigned long
__RPC_FAR *, unsigned char __RPC_FAR *,
VARIANT __RPC_FAR * );
void                      __RPC_USER
VARIANT_UserFree(     unsigned long
__RPC_FAR *, VARIANT __RPC_FAR * );

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

Tpcc_com_ps_i.h

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

/* 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
5.03.0280 */
/* at Thu Dec 13 23:13:08 2001
 */
/* Compiler settings for
.\src\tpcc_com_ps.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32
(32b run), ms_ext, c_ext
    error checks: allocation ref
bounds_check enum stub_data
    VC __declspec() decoration level:
         __declspec(uuid()),
__declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(),
MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if !defined(_M_IA64) &&
!defined(_M_AXP64)

#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,b
6,b7,b8)

#else // !_MIDL_USE_GUIDDEF_

#ifndef __IID_DEFINED__
#define __IID_DEFINED__
```

Appendix A:
Source Code

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

#endif // __IID_DEFINED__

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

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

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
IID_ITPCC,0xFEEE6AA2,0x84B1,0x11d2,0xBA,0
x47,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif



#endif /* !defined(_M_IA64) &&
!defined(_M_AXP64)*/



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

/* 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
5.03.0280 */
/* at Thu Dec 13 23:13:08 2001
 */
/* Compiler settings for
.\src\tpcc_com_ps.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win64
(32b run,appending), 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_AXP64)

#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,b
6,b7,b8)

#else // !_MIDL_USE_GUIDDEF_

#ifndef __IID_DEFINED__
#define __IID_DEFINED__

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

#endif // __IID_DEFINED__

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

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

#endif !_MIDL_USE_GUIDDEF_

MIDL_DEFINE_GUID(IID,
IID_ITPCC,0xFEEE6AA2,0x84B1,0x11d2,0xBA,0
x47,0x00,0xC0,0x4F,0xBF,0xE0,0x8B);

#undef MIDL_DEFINE_GUID

#ifdef __cplusplus
}
#endif
```

Appendix A: Source Code

```
#endif /* defined(_M_IA64) ||
defined(_M_AXP64)*/
```

Tpcc_com_ps_c.h

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

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


 /* File created by MIDL compiler version
5.03.0280 */
/* at Thu Dec 13 23:13:08 2001
 */
/* Compiler settings for
.\src\tpcc_com_ps.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32
(32b run), ms_ext, c_ext
    error checks: allocation ref
bounds_check enum stub_data
    VC __declspec() decoration level:
         __declspec(uuid()),
__declspec(selectany),
__declspec(novtable)
         DECLSPEC_UUID(),
MIDL_INTERFACE()
*/
//@@MIDL_FILE_HEADING(  )

#if !defined(_M_IA64) &&
!defined(_M_AXP64)
#define USE_STUBLESS_PROXY


/* verify that the <rpcproxy.h> version
is high enough to compile this file*/
#ifndef __REDQ_RPCPROXY_H_VERSION__
#define __REQUIRED_RPCPROXY_H_VERSION__
440
#endif


#include "rpcproxy.h"
#ifndef __RPCPROXY_H_VERSION__
#error this stub requires an updated
version of <rpcproxy.h>
#endif // __RPCPROXY_H_VERSION__


#include "tpcc_com_ps.h"

#define TYPE_FORMAT_STRING_SIZE   997
#define PROC_FORMAT_STRING_SIZE   193
#define TRANSMIT_AS_TABLE_SIZE    0
#define WIRE_MARSHAL_TABLE_SIZE   1

typedef struct _MIDL_TYPE_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[
TYPE_FORMAT_STRING_SIZE ];
    } MIDL_TYPE_FORMAT_STRING;

typedef struct _MIDL_PROC_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[
PROC_FORMAT_STRING_SIZE ];
    } MIDL_PROC_FORMAT_STRING;


extern const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString;
extern const MIDL_PROC_FORMAT_STRING
__MIDL_ProcFormatString;


/* Standard interface:
__MIDL_itf_tpcc_com_ps_0000, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00}} */


/* Object interface: IUnknown, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46}} */


/* Object interface: ITPCC, ver. 0.0,

GUID={0xFEEE6AA2,0x84B1,0x11d2,{0xBA,0x47
,0x00,0xC0,0x4F,0xBF,0xE0,0x8B}} */


extern const MIDL_STUB_DESC
Object_StubDesc;


extern const MIDL_SERVER_INFO
ITPCC_ServerInfo;

#pragma code_seg(".orpc")
static const unsigned short
ITPCC_FormatStringOffsetTable[] =
    {
    0,
    34,
    68,
    102,
    136,
    170
    };

static const MIDL_SERVER_INFO
ITPCC_ServerInfo =
    {
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &ITPCC_FormatStringOffsetTable[-3],

    0,
    0,
    0,
    0
    };

static const MIDL_STUBLESS_PROXY_INFO
ITPCC_ProxyInfo =
    {
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &ITPCC_FormatStringOffsetTable[-3],
```

```
    0,
    0,
    0
    };

CINTERFACE_PROXY_VTABLE(9)
_ITPCCProxyVtbl =
{
    &ITPCC_ProxyInfo,
    &IID_ITPCC,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    (void *)-1 /* ITPCC::NewOrder */ ,
    (void *)-1 /* ITPCC::Payment */ ,
    (void *)-1 /* ITPCC::Delivery */ ,
    (void *)-1 /* ITPCC::StockLevel */ ,
    (void *)-1 /* ITPCC::OrderStatus */ ,
    (void *)-1 /* ITPCC::CallSetComplete
*/
};

const CInterfaceStubVtbl _ITPCCStubVtbl =
{
    &IID_ITPCC,
    &ITPCC_ServerInfo,
    9,
    0, /* pure interpreted */
    CStdStubBuffer_METHODS
};

extern const
USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[
WIRE_MARSHAL_TABLE_SIZE ];

static const MIDL_STUB_DESC
Object_StubDesc =
    {
    0,
    NdrOleAllocate,
    NdrOleFree,
    0,
    0,
    0,
    0,
    0,
    __MIDL_TypeFormatString.Format,
    1, /* -error bounds_check flag */
    0x20000, /* Ndr library version */
    0,
    0x5030118, /* MIDL Version 5.3.280 */
    0,
    UserMarshalRoutines,
    0,  /* notify & notify_flag routine
table */
    0x1, /* MIDL flag */
    0,  /* Reserved3 */
    0,  /* Reserved4 */
    0   /* Reserved5 */
    };

#pragma data_seg(".rdata")

static const
USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[
WIRE_MARSHAL_TABLE_SIZE ] =
        {
            {
            VARIANT_UserSize
            ,VARIANT_UserMarshal
            ,VARIANT_UserUnmarshal
            ,VARIANT_UserFree
            }

        };


#if !defined(__RPC_WIN32__)
#error  Invalid build platform for this
stub.
#endif

#if !(TARGET_IS_NT40_OR_LATER)
#error You need a Windows NT 4.0 or later
to run this stub because it uses these
features:
#error   -Oif or -Oicf, [wire_marshal] or
[user_marshal] attribute.
#error However, your C/C++ compilation
flags indicate you intend to run this app
on earlier systems.
#error This app will die there with the
RPC_X_WRONG_STUB_VERSION error.
#endif


static const MIDL_PROC_FORMAT_STRING
__MIDL_ProcFormatString =
    {
        0,
        {

        /* Procedure NewOrder */

                        0x33,         /*
FC_AUTO_HANDLE */
                        0x6c,         /*
Old Flags:  object, Oi2 */
/*  2 */        NdrFcLong( 0x0 ),     /*
0 */
/*  6 */        NdrFcShort( 0x3 ),    /*
3 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/*  8 */        NdrFcShort( 0x1c ),   /*
x86 Stack size/offset = 28 */
#else
                        NdrFcShort( 0x20
),      /*  MIPS Stack size/offset = 32
*/
#endif
#else
                        NdrFcShort( 0x20
),      /* PPC Stack size/offset = 32 */
#endif
#else
                        NdrFcShort( 0x28
),      /* Alpha Stack size/offset = 40
*/
#endif
/* 10 */        NdrFcShort( 0x0 ),    /*
0 */
/* 12 */        NdrFcShort( 0x8 ),    /*
8 */
```

```
/* 14 */	0x7,		/* Oi2 
Flags:  srv must size, clt must size, has 
return, */
			0x3,		/* 
3 */

	/* Parameter txn_in */

/* 16 */	NdrFcShort( 0x8b ),	/* 
Flags:  must size, must free, in, by val, 
*/
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 18 */	NdrFcShort( 0x4 ),	/* 
x86 Stack size/offset = 4 */
#else
			NdrFcShort( 0x8 ),	
	/*  MIPS Stack size/offset = 8 */
#endif
#else
			NdrFcShort( 0x8 ),	
	/* PPC Stack size/offset = 8 */
#endif
#else
			NdrFcShort( 0x8 ),	
	/* Alpha Stack size/offset = 8 */
#endif
/* 20 */	NdrFcShort( 0x3c8 ),	/* 
Type Offset=968 */

	/* Parameter txn_out */

/* 22 */	NdrFcShort( 0x4113 ),	/* 
Flags:  must size, must free, out, simple 
ref, srv alloc size=16 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 24 */	NdrFcShort( 0x14 ),	/* 
x86 Stack size/offset = 20 */
#else
			NdrFcShort( 0x18 
),	/*  MIPS Stack size/offset = 24 
*/
#endif
#else
			NdrFcShort( 0x18 
),	/* PPC Stack size/offset = 24 */
#endif
#else
			NdrFcShort( 0x18 
),	/* Alpha Stack size/offset = 24 
*/
#endif
/* 26 */	NdrFcShort( 0x3da ),	/* 
Type Offset=986 */

	/* Return value */

/* 28 */	NdrFcShort( 0x70 ),	/* 
Flags:  out, return, base type, */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 30 */	NdrFcShort( 0x18 ),	/* 
x86 Stack size/offset = 24 */
#else
			NdrFcShort( 0x1c 
),	/*  MIPS Stack size/offset = 28 
*/
#endif
#else
			NdrFcShort( 0x1c 
),	/* PPC Stack size/offset = 28 */
#endif
#else
			NdrFcShort( 0x20 
),	/* Alpha Stack size/offset = 32 
*/
#endif
/* 32 */	0x8,		/* FC_LONG 
*/
			0x0,		/* 
0 */

	/* Procedure Payment */

/* 34 */	0x33,		/* 
FC_AUTO_HANDLE */
			0x6c,		/* 
Old Flags:  object, Oi2 */
/* 36 */	NdrFcLong( 0x0 ),	/* 
0 */
/* 40 */	NdrFcShort( 0x4 ),	/* 
4 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 42 */	NdrFcShort( 0x1c ),	/* 
x86 Stack size/offset = 28 */
#else
			NdrFcShort( 0x20 
),	/*  MIPS Stack size/offset = 32 
*/
#endif
#else
			NdrFcShort( 0x20 
),	/* PPC Stack size/offset = 32 */
#endif
#else
			NdrFcShort( 0x28 
),	/* Alpha Stack size/offset = 40 
*/
#endif
/* 44 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 46 */	NdrFcShort( 0x8 ),	/* 
8 */
/* 48 */	0x7,		/* Oi2 
Flags:  srv must size, clt must size, has 
return, */
			0x3,		/* 
3 */

	/* Parameter txn_in */

/* 50 */	NdrFcShort( 0x8b ),	/* 
Flags:  must size, must free, in, by val, 
*/
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 52 */	NdrFcShort( 0x4 ),	/* 
x86 Stack size/offset = 4 */
#else
			NdrFcShort( 0x8 ),	
	/*  MIPS Stack size/offset = 8 */
```

```
#endif
#else
                        NdrFcShort( 0x8 ),
        /* PPC Stack size/offset = 8 */
#endif
#else
                        NdrFcShort( 0x8 ),
        /* Alpha Stack size/offset = 8 */
#endif
/* 54 */        NdrFcShort( 0x3c8 ),   /*
Type Offset=968 */

        /* Parameter txn_out */

/* 56 */        NdrFcShort( 0x4113 ),  /*
Flags:  must size, must free, out, simple
ref, srv alloc size=16 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 58 */        NdrFcShort( 0x14 ),    /*
x86 Stack size/offset = 20 */
#else
                        NdrFcShort( 0x18
),      /*  MIPS Stack size/offset = 24
*/
#endif
#else
                        NdrFcShort( 0x18
),      /* PPC Stack size/offset = 24 */
#endif
#else
                        NdrFcShort( 0x18
),      /* Alpha Stack size/offset = 24
*/
#endif
/* 60 */        NdrFcShort( 0x3da ),   /*
Type Offset=986 */

        /* Return value */

/* 62 */        NdrFcShort( 0x70 ),    /*
Flags:  out, return, base type, */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 64 */        NdrFcShort( 0x18 ),    /*
x86 Stack size/offset = 24 */
#else
                        NdrFcShort( 0x1c
),      /*  MIPS Stack size/offset = 28
*/
#endif
#else
                        NdrFcShort( 0x1c
),      /* PPC Stack size/offset = 28 */
#endif
#else
                        NdrFcShort( 0x20
),      /* Alpha Stack size/offset = 32
*/
#endif
/* 66 */        0x8,            /* FC_LONG
*/

                        0x0,            /*
0 */

        /* Procedure Delivery */

/* 68 */        0x33,           /*
FC_AUTO_HANDLE */
                        0x6c,           /*
Old Flags:  object, Oi2 */
/* 70 */        NdrFcLong( 0x0 ),      /*
0 */
/* 74 */        NdrFcShort( 0x5 ),     /*
5 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 76 */        NdrFcShort( 0x1c ),    /*
x86 Stack size/offset = 28 */
#else
                        NdrFcShort( 0x20
),      /*  MIPS Stack size/offset = 32
*/
#endif
#else
                        NdrFcShort( 0x20
),      /* PPC Stack size/offset = 32 */
#endif
#else
                        NdrFcShort( 0x28
),      /* Alpha Stack size/offset = 40
*/
#endif
/* 78 */        NdrFcShort( 0x0 ),     /*
0 */
/* 80 */        NdrFcShort( 0x8 ),     /*
8 */
/* 82 */        0x7,            /* Oi2
Flags:  srv must size, clt must size, has
return, */
                        0x3,            /*
3 */

        /* Parameter txn_in */

/* 84 */        NdrFcShort( 0x8b ),    /*
Flags:  must size, must free, in, by val,
*/
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 86 */        NdrFcShort( 0x4 ),     /*
x86 Stack size/offset = 4 */
#else
                        NdrFcShort( 0x8 ),
        /*  MIPS Stack size/offset = 8 */
#endif
#else
                        NdrFcShort( 0x8 ),
        /* PPC Stack size/offset = 8 */
#endif
#else
                        NdrFcShort( 0x8 ),
        /* Alpha Stack size/offset = 8 */
#endif
/* 88 */        NdrFcShort( 0x3c8 ),   /*
Type Offset=968 */

        /* Parameter txn_out */

/* 90 */        NdrFcShort( 0x4113 ),  /*
Flags:  must size, must free, out, simple
ref, srv alloc size=16 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
```

Appendix A:
Source Code

```
/* 92 */	NdrFcShort( 0x14 ),	/* 
x86 Stack size/offset = 20 */
#else
			NdrFcShort( 0x18 
),	/*  MIPS Stack size/offset = 24 
*/
#endif
#else
			NdrFcShort( 0x18 
),	/* PPC Stack size/offset = 24 */
#endif
#else
			NdrFcShort( 0x18 
),	/* Alpha Stack size/offset = 24 
*/
#endif
/* 94 */	NdrFcShort( 0x3da ),	/* 
Type Offset=986 */

	/* Return value */

/* 96 */	NdrFcShort( 0x70 ),	/* 
Flags:  out, return, base type, */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 98 */	NdrFcShort( 0x18 ),	/* 
x86 Stack size/offset = 24 */
#else
			NdrFcShort( 0x1c 
),	/*  MIPS Stack size/offset = 28 
*/
#endif
#else
			NdrFcShort( 0x1c 
),	/* PPC Stack size/offset = 28 */
#endif
#else
			NdrFcShort( 0x20 
),	/* Alpha Stack size/offset = 32 
*/
#endif
/* 100 */	0x8,		/* FC_LONG 
*/
			0x0,		/* 
0 */

	/* Procedure StockLevel */

/* 102 */	0x33,		/* 
FC_AUTO_HANDLE */
			0x6c,		/* 
Old Flags:  object, Oi2 */
/* 104 */	NdrFcLong( 0x0 ),	/* 
0 */
/* 108 */	NdrFcShort( 0x6 ),	/* 
6 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 110 */	NdrFcShort( 0x1c ),	/* 
x86 Stack size/offset = 28 */
#else
			NdrFcShort( 0x20 
),	/*  MIPS Stack size/offset = 32 
*/
#endif
#else
			NdrFcShort( 0x20 
),	/* PPC Stack size/offset = 32 */
#endif
#else
			NdrFcShort( 0x28 
),	/* Alpha Stack size/offset = 40 
*/
#endif
/* 112 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 114 */	NdrFcShort( 0x8 ),	/* 
8 */
/* 116 */	0x7,		/* Oi2 
Flags:  srv must size, clt must size, has 
return, */
			0x3,		/* 
3 */

	/* Parameter txn_in */

/* 118 */	NdrFcShort( 0x8b ),	/* 
Flags:  must size, must free, in, by val, 
*/
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 120 */	NdrFcShort( 0x4 ),	/* 
x86 Stack size/offset = 4 */
#else
			NdrFcShort( 0x8 ),
	/*  MIPS Stack size/offset = 8 */
#endif
#else
			NdrFcShort( 0x8 ),
	/* PPC Stack size/offset = 8 */
#endif
#else
			NdrFcShort( 0x8 ),
	/* Alpha Stack size/offset = 8 */
#endif
/* 122 */	NdrFcShort( 0x3c8 ),	/* 
Type Offset=968 */

	/* Parameter txn_out */

/* 124 */	NdrFcShort( 0x4113 ),	/* 
Flags:  must size, must free, out, simple 
ref, srv alloc size=16 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 126 */	NdrFcShort( 0x14 ),	/* 
x86 Stack size/offset = 20 */
#else
			NdrFcShort( 0x18 
),	/*  MIPS Stack size/offset = 24 
*/
#endif
#else
			NdrFcShort( 0x18 
),	/* PPC Stack size/offset = 24 */
#endif
#else
			NdrFcShort( 0x18 
),	/* Alpha Stack size/offset = 24 
*/
#endif
/* 128 */	NdrFcShort( 0x3da ),	/* 
Type Offset=986 */

	/* Return value */
```

```
/* 130 */	NdrFcShort( 0x70 ),	/*
Flags:  out, return, base type, */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 132 */	NdrFcShort( 0x18 ),	/*
x86 Stack size/offset = 24 */
#else
			NdrFcShort( 0x1c
),	/*  MIPS Stack size/offset = 28
*/
#endif
#else
			NdrFcShort( 0x1c
),	/* PPC Stack size/offset = 28 */
#endif
#else
			NdrFcShort( 0x20
),	/* Alpha Stack size/offset = 32
*/
#endif
/* 134 */	0x8,		/* FC_LONG
*/
			0x0,		/*
0 */

	/* Procedure OrderStatus */

/* 136 */	0x33,		/*
FC_AUTO_HANDLE */
			0x6c,		/*
Old Flags:  object, Oi2 */
/* 138 */	NdrFcLong( 0x0 ),	/*
0 */
/* 142 */	NdrFcShort( 0x7 ),	/*
7 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 144 */	NdrFcShort( 0x1c ),	/*
x86 Stack size/offset = 28 */
#else
			NdrFcShort( 0x20
),	/*  MIPS Stack size/offset = 32
*/
#endif
#else
			NdrFcShort( 0x20
),	/* PPC Stack size/offset = 32 */
#endif
#else
			NdrFcShort( 0x28
),	/* Alpha Stack size/offset = 40
*/
#endif
/* 146 */	NdrFcShort( 0x0 ),	/*
0 */
/* 148 */	NdrFcShort( 0x8 ),	/*
8 */
/* 150 */	0x7,		/* Oi2
Flags:  srv must size, clt must size, has
return, */
			0x3,		/*
3 */

	/* Parameter txn_in */

/* 152 */	NdrFcShort( 0x8b ),	/*
Flags:  must size, must free, in, by val,
*/
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 154 */	NdrFcShort( 0x4 ),	/*
x86 Stack size/offset = 4 */
#else
			NdrFcShort( 0x8 ),
	/*  MIPS Stack size/offset = 8 */
#endif
#else
			NdrFcShort( 0x8 ),
	/* PPC Stack size/offset = 8 */
#endif
#else
			NdrFcShort( 0x8 ),
	/* Alpha Stack size/offset = 8 */
#endif
/* 156 */	NdrFcShort( 0x3c8 ),	/*
Type Offset=968 */

	/* Parameter txn_out */

/* 158 */	NdrFcShort( 0x4113 ),	/*
Flags:  must size, must free, out, simple
ref, srv alloc size=16 */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 160 */	NdrFcShort( 0x14 ),	/*
x86 Stack size/offset = 20 */
#else
			NdrFcShort( 0x18
),	/*  MIPS Stack size/offset = 24
*/
#endif
#else
			NdrFcShort( 0x18
),	/* PPC Stack size/offset = 24 */
#endif
#else
			NdrFcShort( 0x18
),	/* Alpha Stack size/offset = 24
*/
#endif
/* 162 */	NdrFcShort( 0x3da ),	/*
Type Offset=986 */

	/* Return value */

/* 164 */	NdrFcShort( 0x70 ),	/*
Flags:  out, return, base type, */
#ifndef _ALPHA_
#ifndef _PPC_
#if !defined(_MIPS_)
/* 166 */	NdrFcShort( 0x18 ),	/*
x86 Stack size/offset = 24 */
#else
			NdrFcShort( 0x1c
),	/*  MIPS Stack size/offset = 28
*/
#endif
#else
			NdrFcShort( 0x1c
),	/* PPC Stack size/offset = 28 */
#endif
#else
```

```
                        NdrFcShort( 0x20
),      /* Alpha Stack size/offset = 32
*/
#endif
/* 168 */       0x8,            /* FC_LONG
*/
                        0x0,            /*
0 */

        /* Procedure CallSetComplete */

/* 170 */       0x33,           /*
FC_AUTO_HANDLE */
                        0x6c,           /*
Old Flags:  object, Oi2 */
/* 172 */       NdrFcLong( 0x0 ),       /*
0 */
/* 176 */       NdrFcShort( 0x8 ),      /*
8 */
#ifndef _ALPHA_
/* 178 */       NdrFcShort( 0x8 ),      /*
x86, MIPS, PPC Stack size/offset = 8 */
#else
                        NdrFcShort( 0x10
),      /* Alpha Stack size/offset = 16
*/
#endif
/* 180 */       NdrFcShort( 0x0 ),      /*
0 */
/* 182 */       NdrFcShort( 0x8 ),      /*
8 */
/* 184 */       0x4,            /* Oi2
Flags:  has return, */
                        0x1,            /*
1 */

        /* Return value */

/* 186 */       NdrFcShort( 0x70 ),     /*
Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 188 */       NdrFcShort( 0x4 ),      /*
x86, MIPS, PPC Stack size/offset = 4 */
#else
                        NdrFcShort( 0x8 ),
        /* Alpha Stack size/offset = 8 */
#endif
/* 190 */       0x8,            /* FC_LONG
*/
                        0x0,            /*
0 */

                        0x0
        }
    };


static const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString =
    {
        0,
        {
                        NdrFcShort( 0x0 ),
        /* 0 */
/*  2 */
                        0x12, 0x0,      /*
FC_UP */
/*  4 */        NdrFcShort( 0x3b0 ),    /*
Offset= 944 (948) */
/*  6 */
                        0x2b,           /*
FC_NON_ENCAPSULATED_UNION */
                        0x9,            /*
FC_ULONG */
/*  8 */        0x7,            /* Corr
desc: FC_USHORT */
                        0x0,            /*
*/
/* 10 */        NdrFcShort( 0xfff8 ),   /*
-8 */
/* 12 */        NdrFcShort( 0x2 ),      /*
Offset= 2 (14) */
/* 14 */        NdrFcShort( 0x10 ),     /*
16 */
/* 16 */        NdrFcShort( 0x2b ),     /*
43 */
/* 18 */        NdrFcLong( 0x3 ),       /*
3 */
/* 22 */        NdrFcShort( 0x8008 ),   /*
Simple arm type: FC_LONG */
/* 24 */        NdrFcLong( 0x11 ),      /*
17 */
/* 28 */        NdrFcShort( 0x8001 ),   /*
Simple arm type: FC_BYTE */
/* 30 */        NdrFcLong( 0x2 ),       /*
2 */
/* 34 */        NdrFcShort( 0x8006 ),   /*
Simple arm type: FC_SHORT */
/* 36 */        NdrFcLong( 0x4 ),       /*
4 */
/* 40 */        NdrFcShort( 0x800a ),   /*
Simple arm type: FC_FLOAT */
/* 42 */        NdrFcLong( 0x5 ),       /*
5 */
/* 46 */        NdrFcShort( 0x800c ),   /*
Simple arm type: FC_DOUBLE */
/* 48 */        NdrFcLong( 0xb ),       /*
11 */
/* 52 */        NdrFcShort( 0x8006 ),   /*
Simple arm type: FC_SHORT */
/* 54 */        NdrFcLong( 0xa ),       /*
10 */
/* 58 */        NdrFcShort( 0x8008 ),   /*
Simple arm type: FC_LONG */
/* 60 */        NdrFcLong( 0x6 ),       /*
6 */
/* 64 */        NdrFcShort( 0xd6 ),     /*
Offset= 214 (278) */
/* 66 */        NdrFcLong( 0x7 ),       /*
7 */
/* 70 */        NdrFcShort( 0x800c ),   /*
Simple arm type: FC_DOUBLE */
/* 72 */        NdrFcLong( 0x8 ),       /*
8 */
/* 76 */        NdrFcShort( 0xd0 ),     /*
Offset= 208 (284) */
/* 78 */        NdrFcLong( 0xd ),       /*
13 */
/* 82 */        NdrFcShort( 0xe2 ),     /*
Offset= 226 (308) */
/* 84 */        NdrFcLong( 0x9 ),       /*
9 */
/* 88 */        NdrFcShort( 0xee ),     /*
Offset= 238 (326) */
/* 90 */        NdrFcLong( 0x2000 ),    /*
8192 */
/* 94 */        NdrFcShort( 0xfa ),     /*
Offset= 250 (344) */
/* 96 */        NdrFcLong( 0x24 ),      /*
36 */
```

```
/* 100 */      NdrFcShort( 0x308 ),   /*
Offset= 776 (876) */
/* 102 */      NdrFcLong( 0x4024 ),   /*
16420 */
/* 106 */      NdrFcShort( 0x302 ),   /*
Offset= 770 (876) */
/* 108 */      NdrFcLong( 0x4011 ),   /*
16401 */
/* 112 */      NdrFcShort( 0x300 ),   /*
Offset= 768 (880) */
/* 114 */      NdrFcLong( 0x4002 ),   /*
16386 */
/* 118 */      NdrFcShort( 0x2fe ),   /*
Offset= 766 (884) */
/* 120 */      NdrFcLong( 0x4003 ),   /*
16387 */
/* 124 */      NdrFcShort( 0x2fc ),   /*
Offset= 764 (888) */
/* 126 */      NdrFcLong( 0x4004 ),   /*
16388 */
/* 130 */      NdrFcShort( 0x2fa ),   /*
Offset= 762 (892) */
/* 132 */      NdrFcLong( 0x4005 ),   /*
16389 */
/* 136 */      NdrFcShort( 0x2f8 ),   /*
Offset= 760 (896) */
/* 138 */      NdrFcLong( 0x400b ),   /*
16395 */
/* 142 */      NdrFcShort( 0x2e6 ),   /*
Offset= 742 (884) */
/* 144 */      NdrFcLong( 0x400a ),   /*
16394 */
/* 148 */      NdrFcShort( 0x2e4 ),   /*
Offset= 740 (888) */
/* 150 */      NdrFcLong( 0x4006 ),   /*
16390 */
/* 154 */      NdrFcShort( 0x2ea ),   /*
Offset= 746 (900) */
/* 156 */      NdrFcLong( 0x4007 ),   /*
16391 */
/* 160 */      NdrFcShort( 0x2e0 ),   /*
Offset= 736 (896) */
/* 162 */      NdrFcLong( 0x4008 ),   /*
16392 */
/* 166 */      NdrFcShort( 0x2e2 ),   /*
Offset= 738 (904) */
/* 168 */      NdrFcLong( 0x400d ),   /*
16397 */
/* 172 */      NdrFcShort( 0x2e0 ),   /*
Offset= 736 (908) */
/* 174 */      NdrFcLong( 0x4009 ),   /*
16393 */
/* 178 */      NdrFcShort( 0x2de ),   /*
Offset= 734 (912) */
/* 180 */      NdrFcLong( 0x6000 ),   /*
24576 */
/* 184 */      NdrFcShort( 0x2dc ),   /*
Offset= 732 (916) */
/* 186 */      NdrFcLong( 0x400c ),   /*
16396 */
/* 190 */      NdrFcShort( 0x2da ),   /*
Offset= 730 (920) */
/* 192 */      NdrFcLong( 0x10 ),     /*
16 */
/* 196 */      NdrFcShort( 0x8002 ),  /*
Simple arm type: FC_CHAR */
/* 198 */      NdrFcLong( 0x12 ),     /*
18 */
/* 202 */      NdrFcShort( 0x8006 ),  /*
Simple arm type: FC_SHORT */
/* 204 */      NdrFcLong( 0x13 ),     /*
19 */
/* 208 */      NdrFcShort( 0x8008 ),  /*
Simple arm type: FC_LONG */
/* 210 */      NdrFcLong( 0x16 ),     /*
22 */
/* 214 */      NdrFcShort( 0x8008 ),  /*
Simple arm type: FC_LONG */
/* 216 */      NdrFcLong( 0x17 ),     /*
23 */
/* 220 */      NdrFcShort( 0x8008 ),  /*
Simple arm type: FC_LONG */
/* 222 */      NdrFcLong( 0xe ),      /*
14 */
/* 226 */      NdrFcShort( 0x2be ),   /*
Offset= 702 (928) */
/* 228 */      NdrFcLong( 0x400e ),   /*
16398 */
/* 232 */      NdrFcShort( 0x2c4 ),   /*
Offset= 708 (940) */
/* 234 */      NdrFcLong( 0x4010 ),   /*
16400 */
/* 238 */      NdrFcShort( 0x2c2 ),   /*
Offset= 706 (944) */
/* 240 */      NdrFcLong( 0x4012 ),   /*
16402 */
/* 244 */      NdrFcShort( 0x280 ),   /*
Offset= 640 (884) */
/* 246 */      NdrFcLong( 0x4013 ),   /*
16403 */
/* 250 */      NdrFcShort( 0x27e ),   /*
Offset= 638 (888) */
/* 252 */      NdrFcLong( 0x4016 ),   /*
16406 */
/* 256 */      NdrFcShort( 0x278 ),   /*
Offset= 632 (888) */
/* 258 */      NdrFcLong( 0x4017 ),   /*
16407 */
/* 262 */      NdrFcShort( 0x272 ),   /*
Offset= 626 (888) */
/* 264 */      NdrFcLong( 0x0 ),      /*
0 */
/* 268 */      NdrFcShort( 0x0 ),     /*
Offset= 0 (268) */
/* 270 */      NdrFcLong( 0x1 ),      /*
1 */
/* 274 */      NdrFcShort( 0x0 ),     /*
Offset= 0 (274) */
/* 276 */      NdrFcShort( 0xffffffff ),
               /* Offset= -1 (275) */
/* 278 */
                             0x15,          /*
FC_STRUCT */
                             0x7,           /*
7 */
/* 280 */      NdrFcShort( 0x8 ),     /*
8 */
/* 282 */      0xb,           /* FC_HYPER
*/
                             0x5b,          /*
FC_END */
/* 284 */
                             0x12, 0x0,     /*
FC_UP */
/* 286 */      NdrFcShort( 0xc ),     /*
Offset= 12 (298) */
/* 288 */
                             0x1b,          /*
FC_CARRAY */
```

```
			0x1,		/* 
1 */
/* 290 */	NdrFcShort( 0x2 ),	/* 
2 */
/* 292 */	0x9,		/* Corr 
desc: FC_ULONG */
			0x0,		/* 
*/
/* 294 */	NdrFcShort( 0xfffc ),	/* 
-4 */
/* 296 */	0x6,		/* FC_SHORT 
*/
			0x5b,		/* 
FC_END */
/* 298 */
			0x17,		/* 
FC_CSTRUCT */
			0x3,		/* 
3 */
/* 300 */	NdrFcShort( 0x8 ),	/* 
8 */
/* 302 */	NdrFcShort( 0xfffffff2 ),
	/* Offset= -14 (288) */
/* 304 */	0x8,		/* FC_LONG 
*/
			0x8,		/* 
FC_LONG */
/* 306 */	0x5c,		/* FC_PAD 
*/
			0x5b,		/* 
FC_END */
/* 308 */
			0x2f,		/* 
FC_IP */
			0x5a,		/* 
FC_CONSTANT_IID */
/* 310 */	NdrFcLong( 0x0 ),	/* 
0 */
/* 314 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 316 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 318 */	0xc0,		/* 192 */
			0x0,		/* 
0 */
/* 320 */	0x0,		/* 0 */
			0x0,		/* 
0 */
/* 322 */	0x0,		/* 0 */
			0x0,		/* 
0 */
/* 324 */	0x0,		/* 0 */
			0x46,		/* 
70 */
/* 326 */
			0x2f,		/* 
FC_IP */
			0x5a,		/* 
FC_CONSTANT_IID */
/* 328 */	NdrFcLong( 0x20400 ),	/* 
132096 */
/* 332 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 334 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 336 */	0xc0,		/* 192 */
			0x0,		/* 
0 */
/* 338 */	0x0,		/* 0 */
			0x0,		/* 
0 */
/* 340 */	0x0,		/* 0 */
			0x0,		/* 
0 */
/* 342 */	0x0,		/* 0 */
			0x46,		/* 
70 */
/* 344 */
			0x12, 0x10,	/* 
FC_UP [pointer_deref] */
/* 346 */	NdrFcShort( 0x2 ),	/* 
Offset= 2 (348) */
/* 348 */
			0x12, 0x0,	/* 
FC_UP */
/* 350 */	NdrFcShort( 0x1fc ),	/* 
Offset= 508 (858) */
/* 352 */
			0x2a,		/* 
FC_ENCAPSULATED_UNION */
			0x49,		/* 
73 */
/* 354 */	NdrFcShort( 0x18 ),	/* 
24 */
/* 356 */	NdrFcShort( 0xa ),	/* 
10 */
/* 358 */	NdrFcLong( 0x8 ),	/* 
8 */
/* 362 */	NdrFcShort( 0x58 ),	/* 
Offset= 88 (450) */
/* 364 */	NdrFcLong( 0xd ),	/* 
13 */
/* 368 */	NdrFcShort( 0x78 ),	/* 
Offset= 120 (488) */
/* 370 */	NdrFcLong( 0x9 ),	/* 
9 */
/* 374 */	NdrFcShort( 0x94 ),	/* 
Offset= 148 (522) */
/* 376 */	NdrFcLong( 0xc ),	/* 
12 */
/* 380 */	NdrFcShort( 0xbc ),	/* 
Offset= 188 (568) */
/* 382 */	NdrFcLong( 0x24 ),	/* 
36 */
/* 386 */	NdrFcShort( 0x114 ),	/* 
Offset= 276 (662) */
/* 388 */	NdrFcLong( 0x800d ),	/* 
32781 */
/* 392 */	NdrFcShort( 0x130 ),	/* 
Offset= 304 (696) */
/* 394 */	NdrFcLong( 0x10 ),	/* 
16 */
/* 398 */	NdrFcShort( 0x148 ),	/* 
Offset= 328 (726) */
/* 400 */	NdrFcLong( 0x2 ),	/* 
2 */
/* 404 */	NdrFcShort( 0x160 ),	/* 
Offset= 352 (756) */
/* 406 */	NdrFcLong( 0x3 ),	/* 
3 */
/* 410 */	NdrFcShort( 0x178 ),	/* 
Offset= 376 (786) */
/* 412 */	NdrFcLong( 0x14 ),	/* 
20 */
/* 416 */	NdrFcShort( 0x190 ),	/* 
Offset= 400 (816) */
/* 418 */	NdrFcShort( 0xffffffff ),
	/* Offset= -1 (417) */
/* 420 */
```

```
                        0x1b,           /*
FC_CARRAY */
                        0x3,            /*
3 */
/* 422 */       NdrFcShort( 0x4 ),      /*
4 */
/* 424 */       0x19,           /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,            /*
*/
/* 426 */       NdrFcShort( 0x0 ),      /*
0 */
/* 428 */
                        0x4b,           /*
FC_PP */
                        0x5c,           /*
FC_PAD */
/* 430 */
                        0x48,           /*
FC_VARIABLE_REPEAT */
                        0x49,           /*
FC_FIXED_OFFSET */
/* 432 */       NdrFcShort( 0x4 ),      /*
4 */
/* 434 */       NdrFcShort( 0x0 ),      /*
0 */
/* 436 */       NdrFcShort( 0x1 ),      /*
1 */
/* 438 */       NdrFcShort( 0x0 ),      /*
0 */
/* 440 */       NdrFcShort( 0x0 ),      /*
0 */
/* 442 */       0x12, 0x0,      /* FC_UP */
/* 444 */       NdrFcShort( 0xffffff6e ),
        /* Offset= -146 (298) */
/* 446 */
                        0x5b,           /*
FC_END */

                        0x8,            /*
FC_LONG */
/* 448 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 450 */
                        0x16,           /*
FC_PSTRUCT */
                        0x3,            /*
3 */
/* 452 */       NdrFcShort( 0x8 ),      /*
8 */
/* 454 */
                        0x4b,           /*
FC_PP */
                        0x5c,           /*
FC_PAD */
/* 456 */
                        0x46,           /*
FC_NO_REPEAT */
                        0x5c,           /*
FC_PAD */
/* 458 */       NdrFcShort( 0x4 ),      /*
4 */
/* 460 */       NdrFcShort( 0x4 ),      /*
4 */
/* 462 */       0x11, 0x0,      /* FC_RP */
/* 464 */       NdrFcShort( 0xffffffd4 ),
        /* Offset= -44 (420) */
/* 466 */
                        0x5b,           /*
FC_END */

                        0x8,            /*
FC_LONG */
/* 468 */       0x8,            /* FC_LONG
*/
                        0x5b,           /*
FC_END */
/* 470 */
                        0x21,           /*
FC_BOGUS_ARRAY */
                        0x3,            /*
3 */
/* 472 */       NdrFcShort( 0x0 ),      /*
0 */
/* 474 */       0x19,           /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,            /*
*/
/* 476 */       NdrFcShort( 0x0 ),      /*
0 */
/* 478 */       NdrFcLong( 0xffffffff ),
        /* -1 */
/* 482 */       0x4c,           /*
FC_EMBEDDED_COMPLEX */
                        0x0,            /*
0 */
/* 484 */       NdrFcShort( 0xffffff50 ),
        /* Offset= -176 (308) */
/* 486 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 488 */
                        0x1a,           /*
FC_BOGUS_STRUCT */
                        0x3,            /*
3 */
/* 490 */       NdrFcShort( 0x8 ),      /*
8 */
/* 492 */       NdrFcShort( 0x0 ),      /*
0 */
/* 494 */       NdrFcShort( 0x6 ),      /*
Offset= 6 (500) */
/* 496 */       0x8,            /* FC_LONG
*/
                        0x36,           /*
FC_POINTER */
/* 498 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 500 */
                        0x11, 0x0,      /*
FC_RP */
/* 502 */       NdrFcShort( 0xffffffe0 ),
        /* Offset= -32 (470) */
/* 504 */
                        0x21,           /*
FC_BOGUS_ARRAY */
                        0x3,            /*
3 */
/* 506 */       NdrFcShort( 0x0 ),      /*
0 */
/* 508 */       0x19,           /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,            /*
*/
```

```
/* 510 */      NdrFcShort( 0x0 ),     /*
0 */
/* 512 */      NdrFcLong( 0xffffffff ),
        /* -1 */
/* 516 */      0x4c,           /*
FC_EMBEDDED_COMPLEX */
                      0x0,           /*
0 */
/* 518 */      NdrFcShort( 0xffffff40 ),
        /* Offset= -192 (326) */
/* 520 */      0x5c,           /* FC_PAD
*/
                      0x5b,          /*
FC_END */
/* 522 */
                      0x1a,          /*
FC_BOGUS_STRUCT */
                      0x3,           /*
3 */
/* 524 */      NdrFcShort( 0x8 ),     /*
8 */
/* 526 */      NdrFcShort( 0x0 ),     /*
0 */
/* 528 */      NdrFcShort( 0x6 ),     /*
Offset= 6 (534) */
/* 530 */      0x8,            /* FC_LONG
*/
                      0x36,          /*
FC_POINTER */
/* 532 */      0x5c,           /* FC_PAD
*/
                      0x5b,          /*
FC_END */
/* 534 */
                      0x11, 0x0,     /*
FC_RP */
/* 536 */      NdrFcShort( 0xffffffe0 ),
        /* Offset= -32 (504) */
/* 538 */
                      0x1b,          /*
FC_CARRAY */
                      0x3,           /*
3 */
/* 540 */      NdrFcShort( 0x4 ),     /*
4 */
/* 542 */      0x19,           /* Corr
desc:  field pointer, FC_ULONG */
                      0x0,           /*
*/
/* 544 */      NdrFcShort( 0x0 ),     /*
0 */
/* 546 */
                      0x4b,          /*
FC_PP */
                      0x5c,          /*
FC_PAD */
/* 548 */
                      0x48,          /*
FC_VARIABLE_REPEAT */
                      0x49,          /*
FC_FIXED_OFFSET */
/* 550 */      NdrFcShort( 0x4 ),     /*
4 */
/* 552 */      NdrFcShort( 0x0 ),     /*
0 */
/* 554 */      NdrFcShort( 0x1 ),     /*
1 */
/* 556 */      NdrFcShort( 0x0 ),     /*
0 */
/* 558 */      NdrFcShort( 0x0 ),     /*
0 */
/* 560 */      0x12, 0x0,      /* FC_UP */
/* 562 */      NdrFcShort( 0x182 ),   /*
Offset= 386 (948) */
/* 564 */
                      0x5b,          /*
FC_END */

                      0x8,           /*
FC_LONG */
/* 566 */      0x5c,           /* FC_PAD
*/
                      0x5b,          /*
FC_END */
/* 568 */
                      0x1a,          /*
FC_BOGUS_STRUCT */
                      0x3,           /*
3 */
/* 570 */      NdrFcShort( 0x8 ),     /*
8 */
/* 572 */      NdrFcShort( 0x0 ),     /*
0 */
/* 574 */      NdrFcShort( 0x6 ),     /*
Offset= 6 (580) */
/* 576 */      0x8,            /* FC_LONG
*/
                      0x36,          /*
FC_POINTER */
/* 578 */      0x5c,           /* FC_PAD
*/
                      0x5b,          /*
FC_END */
/* 580 */
                      0x11, 0x0,     /*
FC_RP */
/* 582 */      NdrFcShort( 0xffffffd4 ),
        /* Offset= -44 (538) */
/* 584 */
                      0x2f,          /*
FC_IP */
                      0x5a,          /*
FC_CONSTANT_IID */
/* 586 */      NdrFcLong( 0x2f ),     /*
47 */
/* 590 */      NdrFcShort( 0x0 ),     /*
0 */
/* 592 */      NdrFcShort( 0x0 ),     /*
0 */
/* 594 */      0xc0,           /* 192 */
                      0x0,           /*
0 */
/* 596 */      0x0,            /* 0 */
                      0x0,           /*
0 */
/* 598 */      0x0,            /* 0 */
                      0x0,           /*
0 */
/* 600 */      0x0,            /* 0 */
                      0x46,          /*
70 */
/* 602 */
                      0x1b,          /*
FC_CARRAY */
                      0x0,           /*
0 */
/* 604 */      NdrFcShort( 0x1 ),     /*
1 */
```

```
/* 606 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,           /*
*/
/* 608 */      NdrFcShort( 0x4 ),     /*
4 */
/* 610 */      0x1,           /* FC_BYTE
*/
                        0x5b,          /*
FC_END */
/* 612 */
                        0x1a,          /*
FC_BOGUS_STRUCT */
                        0x3,           /*
3 */
/* 614 */      NdrFcShort( 0x10 ),    /*
16 */
/* 616 */      NdrFcShort( 0x0 ),     /*
0 */
/* 618 */      NdrFcShort( 0xa ),     /*
Offset= 10 (628) */
/* 620 */      0x8,           /* FC_LONG
*/
                        0x8,           /*
FC_LONG */
/* 622 */      0x4c,          /*
FC_EMBEDDED_COMPLEX */
                        0x0,           /*
0 */
/* 624 */      NdrFcShort( 0xffffffd8 ),
        /* Offset= -40 (584) */
/* 626 */      0x36,          /*
FC_POINTER */
                        0x5b,          /*
FC_END */
/* 628 */
                        0x12, 0x0,     /*
FC_UP */
/* 630 */      NdrFcShort( 0xffffffe4 ),
        /* Offset= -28 (602) */
/* 632 */
                        0x1b,          /*
FC_CARRAY */
                        0x3,           /*
3 */
/* 634 */      NdrFcShort( 0x4 ),     /*
4 */
/* 636 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,           /*
*/
/* 638 */      NdrFcShort( 0x0 ),     /*
0 */
/* 640 */
                        0x4b,          /*
FC_PP */
                        0x5c,          /*
FC_PAD */
/* 642 */
                        0x48,          /*
FC_VARIABLE_REPEAT */
                        0x49,          /*
FC_FIXED_OFFSET */
/* 644 */      NdrFcShort( 0x4 ),     /*
4 */
/* 646 */      NdrFcShort( 0x0 ),     /*
0 */
/* 648 */      NdrFcShort( 0x1 ),     /*
1 */
/* 650 */      NdrFcShort( 0x0 ),     /*
0 */
/* 652 */      NdrFcShort( 0x0 ),     /*
0 */
/* 654 */      0x12, 0x0,     /* FC_UP */
/* 656 */      NdrFcShort( 0xffffffd4 ),
        /* Offset= -44 (612) */
/* 658 */
                        0x5b,          /*
FC_END */

                        0x8,           /*
FC_LONG */
/* 660 */      0x5c,          /* FC_PAD
*/
                        0x5b,          /*
FC_END */
/* 662 */
                        0x1a,          /*
FC_BOGUS_STRUCT */
                        0x3,           /*
3 */
/* 664 */      NdrFcShort( 0x8 ),     /*
8 */
/* 666 */      NdrFcShort( 0x0 ),     /*
0 */
/* 668 */      NdrFcShort( 0x6 ),     /*
Offset= 6 (674) */
/* 670 */      0x8,           /* FC_LONG
*/
                        0x36,          /*
FC_POINTER */
/* 672 */      0x5c,          /* FC_PAD
*/
                        0x5b,          /*
FC_END */
/* 674 */
                        0x11, 0x0,     /*
FC_RP */
/* 676 */      NdrFcShort( 0xffffffd4 ),
        /* Offset= -44 (632) */
/* 678 */
                        0x1d,          /*
FC_SMFARRAY */
                        0x0,           /*
0 */
/* 680 */      NdrFcShort( 0x8 ),     /*
8 */
/* 682 */      0x1,           /* FC_BYTE
*/
                        0x5b,          /*
FC_END */
/* 684 */
                        0x15,          /*
FC_STRUCT */
                        0x3,           /*
3 */
/* 686 */      NdrFcShort( 0x10 ),    /*
16 */
/* 688 */      0x8,           /* FC_LONG
*/
                        0x6,           /*
FC_SHORT */
/* 690 */      0x6,           /* FC_SHORT
*/
                        0x4c,          /*
FC_EMBEDDED_COMPLEX */
/* 692 */      0x0,           /* 0 */
                        NdrFcShort(
0xfffffff1 ),  /* Offset= -15 (678) */
```

```
                        0x5b,           /*
FC_END */
/* 696 */
                        0x1a,           /*
FC_BOGUS_STRUCT */
                        0x3,            /*
3 */
/* 698 */       NdrFcShort( 0x18 ),     /*
24 */
/* 700 */       NdrFcShort( 0x0 ),      /*
0 */
/* 702 */       NdrFcShort( 0xa ),      /*
Offset= 10 (712) */
/* 704 */       0x8,            /* FC_LONG
*/
                        0x36,           /*
FC_POINTER */
/* 706 */       0x4c,           /*
FC_EMBEDDED_COMPLEX */
                        0x0,            /*
0 */
/* 708 */       NdrFcShort( 0xffffffe8 ),
        /* Offset= -24 (684) */
/* 710 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 712 */
                        0x11, 0x0,      /*
FC_RP */
/* 714 */       NdrFcShort( 0xffffff0c ),
        /* Offset= -244 (470) */
/* 716 */
                        0x1b,           /*
FC_CARRAY */
                        0x0,            /*
0 */
/* 718 */       NdrFcShort( 0x1 ),      /*
1 */
/* 720 */       0x19,           /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,            /*
*/
/* 722 */       NdrFcShort( 0x0 ),      /*
0 */
/* 724 */       0x1,            /* FC_BYTE
*/
                        0x5b,           /*
FC_END */
/* 726 */
                        0x16,           /*
FC_PSTRUCT */
                        0x3,            /*
3 */
/* 728 */       NdrFcShort( 0x8 ),      /*
8 */
/* 730 */
                        0x4b,           /*
FC_PP */
                        0x5c,           /*
FC_PAD */
/* 732 */
                        0x46,           /*
FC_NO_REPEAT */
                        0x5c,           /*
FC_PAD */
/* 734 */       NdrFcShort( 0x4 ),      /*
4 */
/* 736 */       NdrFcShort( 0x4 ),      /*
4 */
/* 738 */       0x12, 0x0,      /* FC_UP */
/* 740 */       NdrFcShort( 0xffffffe8 ),
        /* Offset= -24 (716) */
/* 742 */
                        0x5b,           /*
FC_END */

                        0x8,            /*
FC_LONG */
/* 744 */       0x8,            /* FC_LONG
*/
                        0x5b,           /*
FC_END */
/* 746 */
                        0x1b,           /*
FC_CARRAY */
                        0x1,            /*
1 */
/* 748 */       NdrFcShort( 0x2 ),      /*
2 */
/* 750 */       0x19,           /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,            /*
*/
/* 752 */       NdrFcShort( 0x0 ),      /*
0 */
/* 754 */       0x6,            /* FC_SHORT
*/
                        0x5b,           /*
FC_END */
/* 756 */
                        0x16,           /*
FC_PSTRUCT */
                        0x3,            /*
3 */
/* 758 */       NdrFcShort( 0x8 ),      /*
8 */
/* 760 */
                        0x4b,           /*
FC_PP */
                        0x5c,           /*
FC_PAD */
/* 762 */
                        0x46,           /*
FC_NO_REPEAT */
                        0x5c,           /*
FC_PAD */
/* 764 */       NdrFcShort( 0x4 ),      /*
4 */
/* 766 */       NdrFcShort( 0x4 ),      /*
4 */
/* 768 */       0x12, 0x0,      /* FC_UP */
/* 770 */       NdrFcShort( 0xffffffe8 ),
        /* Offset= -24 (746) */
/* 772 */
                        0x5b,           /*
FC_END */

                        0x8,            /*
FC_LONG */
/* 774 */       0x8,            /* FC_LONG
*/
                        0x5b,           /*
FC_END */
/* 776 */
                        0x1b,           /*
FC_CARRAY */
                        0x3,            /*
3 */
```

```
/* 778 */      NdrFcShort( 0x4 ),     /*
4 */
/* 780 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                      0x0,            /*
*/
/* 782 */      NdrFcShort( 0x0 ),     /*
0 */
/* 784 */      0x8,           /* FC_LONG
*/
                      0x5b,           /*
FC_END */
/* 786 */
                      0x16,           /*
FC_PSTRUCT */
                      0x3,            /*
3 */
/* 788 */      NdrFcShort( 0x8 ),     /*
8 */
/* 790 */
                      0x4b,           /*
FC_PP */
                      0x5c,           /*
FC_PAD */
/* 792 */
                      0x46,           /*
FC_NO_REPEAT */
                      0x5c,           /*
FC_PAD */
/* 794 */      NdrFcShort( 0x4 ),     /*
4 */
/* 796 */      NdrFcShort( 0x4 ),     /*
4 */
/* 798 */      0x12, 0x0,     /* FC_UP */
/* 800 */      NdrFcShort( 0xffffffe8 ),
        /* Offset= -24 (776) */
/* 802 */
                      0x5b,           /*
FC_END */

                      0x8,            /*
FC_LONG */
/* 804 */      0x8,           /* FC_LONG
*/
                      0x5b,           /*
FC_END */
/* 806 */
                      0x1b,           /*
FC_CARRAY */
                      0x7,            /*
7 */
/* 808 */      NdrFcShort( 0x8 ),     /*
8 */
/* 810 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                      0x0,            /*
*/
/* 812 */      NdrFcShort( 0x0 ),     /*
0 */
/* 814 */      0xb,           /* FC_HYPER
*/
                      0x5b,           /*
FC_END */
/* 816 */
                      0x16,           /*
FC_PSTRUCT */
                      0x3,            /*
3 */
/* 818 */      NdrFcShort( 0x8 ),     /*
8 */
/* 820 */
                      0x4b,           /*
FC_PP */
                      0x5c,           /*
FC_PAD */
/* 822 */
                      0x46,           /*
FC_NO_REPEAT */
                      0x5c,           /*
FC_PAD */
/* 824 */      NdrFcShort( 0x4 ),     /*
4 */
/* 826 */      NdrFcShort( 0x4 ),     /*
4 */
/* 828 */      0x12, 0x0,     /* FC_UP */
/* 830 */      NdrFcShort( 0xffffffe8 ),
        /* Offset= -24 (806) */
/* 832 */
                      0x5b,           /*
FC_END */

                      0x8,            /*
FC_LONG */
/* 834 */      0x8,           /* FC_LONG
*/
                      0x5b,           /*
FC_END */
/* 836 */
                      0x15,           /*
FC_STRUCT */
                      0x3,            /*
3 */
/* 838 */      NdrFcShort( 0x8 ),     /*
8 */
/* 840 */      0x8,           /* FC_LONG
*/
                      0x8,            /*
FC_LONG */
/* 842 */      0x5c,          /* FC_PAD
*/
                      0x5b,           /*
FC_END */
/* 844 */
                      0x1b,           /*
FC_CARRAY */
                      0x3,            /*
3 */
/* 846 */      NdrFcShort( 0x8 ),     /*
8 */
/* 848 */      0x7,           /* Corr
desc: FC_USHORT */
                      0x0,            /*
*/
/* 850 */      NdrFcShort( 0xffd8 ),  /*
-40 */
/* 852 */      0x4c,          /*
FC_EMBEDDED_COMPLEX */
                      0x0,            /*
0 */
/* 854 */      NdrFcShort( 0xffffffee ),
        /* Offset= -18 (836) */
/* 856 */      0x5c,          /* FC_PAD
*/
                      0x5b,           /*
FC_END */
/* 858 */
                      0x1a,           /*
FC_BOGUS_STRUCT */
                      0x3,            /*
3 */
```

```
/* 860 */      NdrFcShort( 0x28 ),    /*
40 */
/* 862 */      NdrFcShort( 0xffffffee ),
       /* Offset= -18 (844) */
/* 864 */      NdrFcShort( 0x0 ),     /*
Offset= 0 (864) */
/* 866 */      0x6,           /* FC_SHORT
*/
                      0x6,           /*
FC_SHORT */
/* 868 */      0x38,          /*
FC_ALIGNM4 */
                      0x8,           /*
FC_LONG */
/* 870 */      0x8,           /* FC_LONG
*/
                      0x4c,          /*
FC_EMBEDDED_COMPLEX */
/* 872 */      0x0,           /* 0 */
                      NdrFcShort(
0xfffffdf7 ),  /* Offset= -521 (352) */
                      0x5b,          /*
FC_END */
/* 876 */
                      0x12, 0x0,     /*
FC_UP */
/* 878 */      NdrFcShort( 0xfffffef6 ),
       /* Offset= -266 (612) */
/* 880 */
                      0x12, 0x8,     /*
FC_UP [simple_pointer] */
/* 882 */      0x1,           /* FC_BYTE
*/
                      0x5c,          /*
FC_PAD */
/* 884 */
                      0x12, 0x8,     /*
FC_UP [simple_pointer] */
/* 886 */      0x6,           /* FC_SHORT
*/
                      0x5c,          /*
FC_PAD */
/* 888 */
                      0x12, 0x8,     /*
FC_UP [simple_pointer] */
/* 890 */      0x8,           /* FC_LONG
*/
                      0x5c,          /*
FC_PAD */

/* 892 */
                      0x12, 0x8,     /*
FC_UP [simple_pointer] */
/* 894 */      0xa,           /* FC_FLOAT
*/
                      0x5c,          /*
FC_PAD */
/* 896 */
                      0x12, 0x8,     /*
FC_UP [simple_pointer] */
/* 898 */      0xc,           /*
FC_DOUBLE */
                      0x5c,          /*
FC_PAD */
/* 900 */
                      0x12, 0x0,     /*
FC_UP */
/* 902 */      NdrFcShort( 0xfffffd90 ),
       /* Offset= -624 (278) */
/* 904 */
```

```
                      0x12, 0x10,    /*
FC_UP [pointer_deref] */
/* 906 */      NdrFcShort( 0xfffffd92 ),
       /* Offset= -622 (284) */
/* 908 */
                      0x12, 0x10,    /*
FC_UP [pointer_deref] */
/* 910 */      NdrFcShort( 0xfffffda6 ),
       /* Offset= -602 (308) */
/* 912 */
                      0x12, 0x10,    /*
FC_UP [pointer_deref] */
/* 914 */      NdrFcShort( 0xfffffdb4 ),
       /* Offset= -588 (326) */
/* 916 */
                      0x12, 0x10,    /*
FC_UP [pointer_deref] */
/* 918 */      NdrFcShort( 0xfffffdc2 ),
       /* Offset= -574 (344) */
/* 920 */
                      0x12, 0x10,    /*
FC_UP [pointer_deref] */
/* 922 */      NdrFcShort( 0x2 ),     /*
Offset= 2 (924) */
/* 924 */
                      0x12, 0x0,     /*
FC_UP */
/* 926 */      NdrFcShort( 0x16 ),    /*
Offset= 22 (948) */
/* 928 */
                      0x15,          /*
FC_STRUCT */
                      0x7,           /*
7 */
/* 930 */      NdrFcShort( 0x10 ),    /*
16 */
/* 932 */      0x6,           /* FC_SHORT
*/
                      0x1,           /*
FC_BYTE */
/* 934 */      0x1,           /* FC_BYTE
*/
                      0x38,          /*
FC_ALIGNM4 */
/* 936 */      0x8,           /* FC_LONG
*/
                      0x39,          /*
FC_ALIGNM8 */
/* 938 */      0xb,           /* FC_HYPER
*/
                      0x5b,          /*
FC_END */
/* 940 */
                      0x12, 0x0,     /*
FC_UP */
/* 942 */      NdrFcShort( 0xfffffff2 ),
       /* Offset= -14 (928) */
/* 944 */
                      0x12, 0x8,     /*
FC_UP [simple_pointer] */
/* 946 */      0x2,           /* FC_CHAR
*/
                      0x5c,          /*
FC_PAD */
/* 948 */
                      0x1a,          /*
FC_BOGUS_STRUCT */
                      0x7,           /*
7 */
```

```
/* 950 */      NdrFcShort( 0x20 ),   /*
32 */
/* 952 */      NdrFcShort( 0x0 ),    /*
0 */
/* 954 */      NdrFcShort( 0x0 ),    /*
Offset= 0 (954) */
/* 956 */      0x8,          /* FC_LONG
*/
                       0x8,          /*
FC_LONG */
/* 958 */      0x6,          /* FC_SHORT
*/
                       0x6,          /*
FC_SHORT */
/* 960 */      0x6,          /* FC_SHORT
*/
                       0x6,          /*
FC_SHORT */
/* 962 */      0x4c,         /*
FC_EMBEDDED_COMPLEX */
                       0x0,          /*
0 */
/* 964 */      NdrFcShort( 0xfffffc42 ),
       /* Offset= -958 (6) */
/* 966 */      0x5c,         /* FC_PAD
*/
                       0x5b,         /*
FC_END */
/* 968 */      0xb4,         /*
FC_USER_MARSHAL */
                       0x83,         /*
131 */
/* 970 */      NdrFcShort( 0x0 ),    /*
0 */
/* 972 */      NdrFcShort( 0x10 ),   /*
16 */
/* 974 */      NdrFcShort( 0x0 ),    /*
0 */
/* 976 */      NdrFcShort( 0xfffffc32 ),
       /* Offset= -974 (2) */
/* 978 */
                       0x11, 0x4,    /*
FC_RP [alloced_on_stack] */
/* 980 */      NdrFcShort( 0x6 ),    /*
Offset= 6 (986) */
/* 982 */
                       0x13, 0x0,    /*
FC_OP */
/* 984 */      NdrFcShort( 0xffffffdc ),
       /* Offset= -36 (948) */
/* 986 */      0xb4,         /*
FC_USER_MARSHAL */
                       0x83,         /*
131 */
/* 988 */      NdrFcShort( 0x0 ),    /*
0 */
/* 990 */      NdrFcShort( 0x10 ),   /*
16 */
/* 992 */      NdrFcShort( 0x0 ),    /*
0 */
/* 994 */      NdrFcShort( 0xfffffff4 ),
       /* Offset= -12 (982) */

                       0x0
        }
    };

const CInterfaceProxyVtbl *
_tpcc_com_ps_ProxyVtblList[] =
{
    ( CInterfaceProxyVtbl *)
&_ITPCCProxyVtbl,
    0
};

const CInterfaceStubVtbl *
_tpcc_com_ps_StubVtblList[] =
{
    ( CInterfaceStubVtbl *)
&_ITPCCStubVtbl,
    0
};

PCInterfaceName const
_tpcc_com_ps_InterfaceNamesList[] =
{
    "ITPCC",
    0
};


#define _tpcc_com_ps_CHECK_IID(n)
        IID_GENERIC_CHECK_IID(
_tpcc_com_ps, pIID, n)

int __stdcall _tpcc_com_ps_IID_Lookup(
const IID * pIID, int * pIndex )
{

    if(!_tpcc_com_ps_CHECK_IID(0))
        {
        *pIndex = 0;
        return 1;
        }

    return 0;
}

const ExtendedProxyFileInfo
tpcc_com_ps_ProxyFileInfo =
{
    (PCInterfaceProxyVtblList *) &
_tpcc_com_ps_ProxyVtblList,
    (PCInterfaceStubVtblList *) &
_tpcc_com_ps_StubVtblList,
    (const PCInterfaceName * ) &
_tpcc_com_ps_InterfaceNamesList,
    0, // no delegation
    & _tpcc_com_ps_IID_Lookup,
    1,
    2,
    0, /* table of [async_uuid]
interfaces */
    0, /* Filler1 */
    0, /* Filler2 */
    0  /* Filler3 */
};


#endif /* !defined(_M_IA64) &&
!defined(_M_AXP64)*/


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

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

Appendix A:
Source Code

```
 /* File created by MIDL compiler version
5.03.0280 */
/* at Thu Dec 13 23:13:08 2001
 */
/* Compiler settings for
.\src\tpcc_com_ps.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win64
(32b run,appending), 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_AXP64)
#define USE_STUBLESS_PROXY


/* verify that the <rpcproxy.h> version
is high enough to compile this file*/
#ifndef __REDQ_RPCPROXY_H_VERSION__
#define __REQUIRED_RPCPROXY_H_VERSION__
475
#endif


#include "rpcproxy.h"
#ifndef __RPCPROXY_H_VERSION__
#error this stub requires an updated
version of <rpcproxy.h>
#endif // __RPCPROXY_H_VERSION__


#include "tpcc_com_ps.h"

#define TYPE_FORMAT_STRING_SIZE   979
#define PROC_FORMAT_STRING_SIZE   253
#define TRANSMIT_AS_TABLE_SIZE    0
#define WIRE_MARSHAL_TABLE_SIZE   1

typedef struct _MIDL_TYPE_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[
TYPE_FORMAT_STRING_SIZE ];
    } MIDL_TYPE_FORMAT_STRING;

typedef struct _MIDL_PROC_FORMAT_STRING
    {
    short          Pad;
    unsigned char  Format[
PROC_FORMAT_STRING_SIZE ];
    } MIDL_PROC_FORMAT_STRING;


extern const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString;
extern const MIDL_PROC_FORMAT_STRING
__MIDL_ProcFormatString;


/* Standard interface:
__MIDL_itf_tpcc_com_ps_0000, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0x00,0x00
,0x00,0x00,0x00,0x00,0x00,0x00}} */


/* Object interface: IUnknown, ver. 0.0,

GUID={0x00000000,0x0000,0x0000,{0xC0,0x00
,0x00,0x00,0x00,0x00,0x00,0x46}} */


/* Object interface: ITPCC, ver. 0.0,

GUID={0xFEEE6AA2,0x84B1,0x11d2,{0xBA,0x47
,0x00,0xC0,0x4F,0xBF,0xE0,0x8B}} */


extern const MIDL_STUB_DESC
Object_StubDesc;


extern const MIDL_SERVER_INFO
ITPCC_ServerInfo;

#pragma code_seg(".orpc")
static const unsigned short
ITPCC_FormatStringOffsetTable[] =
    {
    0,
    44,
    88,
    132,
    176,
    220
    };

static const MIDL_SERVER_INFO
ITPCC_ServerInfo =
    {
    &Object_StubDesc,
    0,
    __MIDL_ProcFormatString.Format,
    &ITPCC_FormatStringOffsetTable[-3],
    0,
    0,
    0,
    0
    };

static const MIDL_STUBLESS_PROXY_INFO
ITPCC_ProxyInfo =
    {
    &Object_StubDesc,
    __MIDL_ProcFormatString.Format,
    &ITPCC_FormatStringOffsetTable[-3],
    0,
    0,
    0
    };

CINTERFACE_PROXY_VTABLE(9)
_ITPCCProxyVtbl =
{
    &ITPCC_ProxyInfo,
    &IID_ITPCC,
    IUnknown_QueryInterface_Proxy,
    IUnknown_AddRef_Proxy,
    IUnknown_Release_Proxy ,
    (void *)-1 /* ITPCC::NewOrder */ ,
    (void *)-1 /* ITPCC::Payment */ ,
```

```
    (void *)-1 /* ITPCC::Delivery */ ,
    (void *)-1 /* ITPCC::StockLevel */ ,
    (void *)-1 /* ITPCC::OrderStatus */ ,
    (void *)-1 /* ITPCC::CallSetComplete
*/
};

const CInterfaceStubVtbl _ITPCCStubVtbl =
{
    &IID_ITPCC,
    &ITPCC_ServerInfo,
    9,
    0, /* pure interpreted */
    CStdStubBuffer_METHODS
};

extern const
USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[
WIRE_MARSHAL_TABLE_SIZE ];

static const MIDL_STUB_DESC
Object_StubDesc =
    {
    0,
    NdrOleAllocate,
    NdrOleFree,
    0,
    0,
    0,
    0,
    0,
    __MIDL_TypeFormatString.Format,
    1, /* -error bounds_check flag */
    0x50002, /* Ndr library version */
    0,
    0x5030118, /* MIDL Version 5.3.280 */
    0,
    UserMarshalRoutines,
    0,  /* notify & notify_flag routine
table */
    0x1, /* MIDL flag */
    0,  /* Reserved3 */
    0,  /* Reserved4 */
    0   /* Reserved5 */
    };

#pragma data_seg(".rdata")

static const
USER_MARSHAL_ROUTINE_QUADRUPLE
UserMarshalRoutines[
WIRE_MARSHAL_TABLE_SIZE ] =
        {

            {
            VARIANT_UserSize
            ,VARIANT_UserMarshal
            ,VARIANT_UserUnmarshal
            ,VARIANT_UserFree
            }

        };


#if !defined(__RPC_WIN64__)
#error  Invalid build platform for this
stub.
#endif
static const MIDL_PROC_FORMAT_STRING
__MIDL_ProcFormatString =
    {
        0,
        {

       /* Procedure NewOrder */

                      0x33,         /*
FC_AUTO_HANDLE */
                      0x6c,         /*
Old Flags:  object, Oi2 */
/*  2 */       NdrFcLong( 0x0 ),     /*
0 */
/*  6 */       NdrFcShort( 0x3 ),    /*
3 */
#ifndef _ALPHA_
/*  8 */       NdrFcShort( 0x38 ),   /*
ia64 Stack size/offset = 56 */
#else
                      NdrFcShort( 0x30
),     /* axp64 Stack size/offset = 48
*/
#endif
/* 10 */       NdrFcShort( 0x0 ),    /*
0 */
/* 12 */       NdrFcShort( 0x8 ),    /*
8 */
/* 14 */       0x47,          /* Oi2
Flags:  srv must size, clt must size, has
return, has ext, */
                      0x3,          /*
3 */
/* 16 */       0xa,           /* 10 */
                      0x7,          /*
Ext Flags:  new corr desc, clt corr
check, srv corr check, */
/* 18 */       NdrFcShort( 0x20 ),   /*
32 */
/* 20 */       NdrFcShort( 0x20 ),   /*
32 */
/* 22 */       NdrFcShort( 0x0 ),    /*
0 */
/* 24 */       NdrFcShort( 0x0 ),    /*
0 */

       /* Parameter txn_in */

/* 26 */       NdrFcShort( 0x8b ),   /*
Flags:  must size, must free, in, by val,
*/
#ifndef _ALPHA_
/* 28 */       NdrFcShort( 0x10 ),   /*
ia64 Stack size/offset = 16 */
#else
                      NdrFcShort( 0x8 ),
       /* axp64 Stack size/offset = 8 */
#endif
/* 30 */       NdrFcShort( 0x3b6 ),  /*
Type Offset=950 */

       /* Parameter txn_out */

/* 32 */       NdrFcShort( 0x6113 ), /*
Flags:  must size, must free, out, simple
ref, srv alloc size=24 */
#ifndef _ALPHA_
/* 34 */       NdrFcShort( 0x28 ),   /*
ia64 Stack size/offset = 40 */
#else
```

```
                        NdrFcShort( 0x20
),      /* axp64 Stack size/offset = 32
*/
#endif
/* 36 */        NdrFcShort( 0x3c8 ),   /*
Type Offset=968 */

        /* Return value */

/* 38 */        NdrFcShort( 0x70 ),    /*
Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 40 */        NdrFcShort( 0x30 ),    /*
ia64 Stack size/offset = 48 */
#else
                        NdrFcShort( 0x28
),      /* axp64 Stack size/offset = 40
*/
#endif
/* 42 */        0x8,            /* FC_LONG
*/
                        0x0,            /*
0 */

        /* Procedure Payment */

/* 44 */        0x33,           /*
FC_AUTO_HANDLE */
                        0x6c,           /*
Old Flags:  object, Oi2 */
/* 46 */        NdrFcLong( 0x0 ),      /*
0 */
/* 50 */        NdrFcShort( 0x4 ),     /*
4 */
#ifndef _ALPHA_
/* 52 */        NdrFcShort( 0x38 ),    /*
ia64 Stack size/offset = 56 */
#else
                        NdrFcShort( 0x30
),      /* axp64 Stack size/offset = 48
*/
#endif
/* 54 */        NdrFcShort( 0x0 ),     /*
0 */
/* 56 */        NdrFcShort( 0x8 ),     /*
8 */
/* 58 */        0x47,           /* Oi2
Flags:  srv must size, clt must size, has
return, has ext, */
                        0x3,            /*
3 */
/* 60 */        0xa,            /* 10 */
                        0x7,            /*
Ext Flags:  new corr desc, clt corr
check, srv corr check, */
/* 62 */        NdrFcShort( 0x20 ),    /*
32 */
/* 64 */        NdrFcShort( 0x20 ),    /*
32 */
/* 66 */        NdrFcShort( 0x0 ),     /*
0 */
/* 68 */        NdrFcShort( 0x0 ),     /*
0 */

        /* Parameter txn_in */

/* 70 */        NdrFcShort( 0x8b ),    /*
Flags:  must size, must free, in, by val,
*/
#ifndef _ALPHA_
/* 72 */        NdrFcShort( 0x10 ),    /*
ia64 Stack size/offset = 16 */
#else
                        NdrFcShort( 0x8 ),
        /* axp64 Stack size/offset = 8 */
#endif
/* 74 */        NdrFcShort( 0x3b6 ),   /*
Type Offset=950 */

        /* Parameter txn_out */


/* 76 */        NdrFcShort( 0x6113 ),  /*
Flags:  must size, must free, out, simple
ref, srv alloc size=24 */
#ifndef _ALPHA_
/* 78 */        NdrFcShort( 0x28 ),    /*
ia64 Stack size/offset = 40 */
#else
                        NdrFcShort( 0x20
),      /* axp64 Stack size/offset = 32
*/
#endif
/* 80 */        NdrFcShort( 0x3c8 ),   /*
Type Offset=968 */

        /* Return value */

/* 82 */        NdrFcShort( 0x70 ),    /*
Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 84 */        NdrFcShort( 0x30 ),    /*
ia64 Stack size/offset = 48 */
#else
                        NdrFcShort( 0x28
),      /* axp64 Stack size/offset = 40
*/
#endif
/* 86 */        0x8,            /* FC_LONG
*/
                        0x0,            /*
0 */

        /* Procedure Delivery */

/* 88 */        0x33,           /*
FC_AUTO_HANDLE */
                        0x6c,           /*
Old Flags:  object, Oi2 */
/* 90 */        NdrFcLong( 0x0 ),      /*
0 */
/* 94 */        NdrFcShort( 0x5 ),     /*
5 */
#ifndef _ALPHA_
/* 96 */        NdrFcShort( 0x38 ),    /*
ia64 Stack size/offset = 56 */
#else
                        NdrFcShort( 0x30
),      /* axp64 Stack size/offset = 48
*/
#endif
/* 98 */        NdrFcShort( 0x0 ),     /*
0 */
/* 100 */       NdrFcShort( 0x8 ),     /*
8 */
/* 102 */       0x47,           /* Oi2
Flags:  srv must size, clt must size, has
return, has ext, */
                        0x3,            /*
3 */
```

```
/* 104 */      0xa,           /* 10 */
                        0x7,           /*
Ext Flags:  new corr desc, clt corr
check, srv corr check, */
/* 106 */      NdrFcShort( 0x20 ),   /*
32 */
/* 108 */      NdrFcShort( 0x20 ),   /*
32 */
/* 110 */      NdrFcShort( 0x0 ),    /*
0 */
/* 112 */      NdrFcShort( 0x0 ),    /*
0 */

        /* Parameter txn_in */

/* 114 */      NdrFcShort( 0x8b ),   /*
Flags:  must size, must free, in, by val,
*/
#ifndef _ALPHA_
/* 116 */      NdrFcShort( 0x10 ),   /*
ia64 Stack size/offset = 16 */
#else
                        NdrFcShort( 0x8 ),
        /* axp64 Stack size/offset = 8 */
#endif
/* 118 */      NdrFcShort( 0x3b6 ),  /*
Type Offset=950 */

        /* Parameter txn_out */

/* 120 */      NdrFcShort( 0x6113 ), /*
Flags:  must size, must free, out, simple
ref, srv alloc size=24 */
#ifndef _ALPHA_
/* 122 */      NdrFcShort( 0x28 ),   /*
ia64 Stack size/offset = 40 */
#else
                        NdrFcShort( 0x20
),      /* axp64 Stack size/offset = 32
*/
#endif
/* 124 */      NdrFcShort( 0x3c8 ),  /*
Type Offset=968 */

        /* Return value */

/* 126 */      NdrFcShort( 0x70 ),   /*
Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 128 */      NdrFcShort( 0x30 ),   /*
ia64 Stack size/offset = 48 */
#else
                        NdrFcShort( 0x28
),      /* axp64 Stack size/offset = 40
*/
#endif
/* 130 */      0x8,           /* FC_LONG
*/
                        0x0,           /*
0 */

        /* Procedure StockLevel */

/* 132 */      0x33,          /*
FC_AUTO_HANDLE */
                        0x6c,          /*
Old Flags:  object, Oi2 */
/* 134 */      NdrFcLong( 0x0 ),     /*
0 */
/* 138 */      NdrFcShort( 0x6 ),    /*
6 */
#ifndef _ALPHA_
/* 140 */      NdrFcShort( 0x38 ),   /*
ia64 Stack size/offset = 56 */
#else
                        NdrFcShort( 0x30
),      /* axp64 Stack size/offset = 48
*/
#endif
/* 142 */      NdrFcShort( 0x0 ),    /*
0 */
/* 144 */      NdrFcShort( 0x8 ),    /*
8 */
/* 146 */      0x47,          /* Oi2
Flags:  srv must size, clt must size, has
return, has ext, */
                        0x3,           /*
3 */
/* 148 */      0xa,           /* 10 */
                        0x7,           /*
Ext Flags:  new corr desc, clt corr
check, srv corr check, */
/* 150 */      NdrFcShort( 0x20 ),   /*
32 */
/* 152 */      NdrFcShort( 0x20 ),   /*
32 */
/* 154 */      NdrFcShort( 0x0 ),    /*
0 */
/* 156 */      NdrFcShort( 0x0 ),    /*
0 */

        /* Parameter txn_in */

/* 158 */      NdrFcShort( 0x8b ),   /*
Flags:  must size, must free, in, by val,
*/
#ifndef _ALPHA_
/* 160 */      NdrFcShort( 0x10 ),   /*
ia64 Stack size/offset = 16 */
#else
                        NdrFcShort( 0x8 ),
        /* axp64 Stack size/offset = 8 */
#endif
/* 162 */      NdrFcShort( 0x3b6 ),  /*
Type Offset=950 */

        /* Parameter txn_out */

/* 164 */      NdrFcShort( 0x6113 ), /*
Flags:  must size, must free, out, simple
ref, srv alloc size=24 */
#ifndef _ALPHA_
/* 166 */      NdrFcShort( 0x28 ),   /*
ia64 Stack size/offset = 40 */
#else
                        NdrFcShort( 0x20
),      /* axp64 Stack size/offset = 32
*/
#endif
/* 168 */      NdrFcShort( 0x3c8 ),  /*
Type Offset=968 */

        /* Return value */

/* 170 */      NdrFcShort( 0x70 ),   /*
Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 172 */      NdrFcShort( 0x30 ),   /*
ia64 Stack size/offset = 48 */
```

```
#else
                        NdrFcShort( 0x28
),      /* axp64 Stack size/offset = 40
*/
#endif
/* 174 */      0x8,           /* FC_LONG
*/
                        0x0,           /*
0 */

       /* Procedure OrderStatus */

/* 176 */      0x33,          /*
FC_AUTO_HANDLE */
                        0x6c,          /*
Old Flags:  object, Oi2 */
/* 178 */      NdrFcLong( 0x0 ),      /*
0 */
/* 182 */      NdrFcShort( 0x7 ),     /*
7 */
#ifndef _ALPHA_
/* 184 */      NdrFcShort( 0x38 ),    /*
ia64 Stack size/offset = 56 */
#else
                        NdrFcShort( 0x30
),      /* axp64 Stack size/offset = 48
*/
#endif
/* 186 */      NdrFcShort( 0x0 ),     /*
0 */
/* 188 */      NdrFcShort( 0x8 ),     /*
8 */
/* 190 */      0x47,          /* Oi2
Flags:  srv must size, clt must size, has
return, has ext, */
                        0x3,           /*
3 */

/* 192 */      0xa,           /* 10 */
                        0x7,           /*
Ext Flags:  new corr desc, clt corr
check, srv corr check, */
/* 194 */      NdrFcShort( 0x20 ),    /*
32 */
/* 196 */      NdrFcShort( 0x20 ),    /*
32 */
/* 198 */      NdrFcShort( 0x0 ),     /*
0 */
/* 200 */      NdrFcShort( 0x0 ),     /*
0 */

       /* Parameter txn_in */

/* 202 */      NdrFcShort( 0x8b ),    /*
Flags:  must size, must free, in, by val,
*/
#ifndef _ALPHA_
/* 204 */      NdrFcShort( 0x10 ),    /*
ia64 Stack size/offset = 16 */
#else
                        NdrFcShort( 0x8 ),
       /* axp64 Stack size/offset = 8 */
#endif
/* 206 */      NdrFcShort( 0x3b6 ),   /*
Type Offset=950 */

       /* Parameter txn_out */

/* 208 */      NdrFcShort( 0x6113 ),  /*
Flags:  must size, must free, out, simple
ref, srv alloc size=24 */
#ifndef _ALPHA_
/* 210 */      NdrFcShort( 0x28 ),    /*
ia64 Stack size/offset = 40 */
#else
                        NdrFcShort( 0x20
),      /* axp64 Stack size/offset = 32
*/
#endif
/* 212 */      NdrFcShort( 0x3c8 ),   /*
Type Offset=968 */

       /* Return value */

/* 214 */      NdrFcShort( 0x70 ),    /*
Flags:  out, return, base type, */
#ifndef _ALPHA_
/* 216 */      NdrFcShort( 0x30 ),    /*
ia64 Stack size/offset = 48 */
#else
                        NdrFcShort( 0x28
),      /* axp64 Stack size/offset = 40
*/
#endif
/* 218 */      0x8,           /* FC_LONG
*/
                        0x0,           /*
0 */

       /* Procedure CallSetComplete */

/* 220 */      0x33,          /*
FC_AUTO_HANDLE */
                        0x6c,          /*
Old Flags:  object, Oi2 */
/* 222 */      NdrFcLong( 0x0 ),      /*
0 */
/* 226 */      NdrFcShort( 0x8 ),     /*
8 */
/* 228 */      NdrFcShort( 0x10 ),    /*
ia64, axp64 Stack size/offset = 16 */
/* 230 */      NdrFcShort( 0x0 ),     /*
0 */
/* 232 */      NdrFcShort( 0x8 ),     /*
8 */
/* 234 */      0x44,          /* Oi2
Flags:  has return, has ext, */
                        0x1,           /*
1 */
/* 236 */      0xa,           /* 10 */
                        0x1,           /*
Ext Flags:  new corr desc, */
/* 238 */      NdrFcShort( 0x0 ),     /*
0 */
/* 240 */      NdrFcShort( 0x0 ),     /*
0 */
/* 242 */      NdrFcShort( 0x0 ),     /*
0 */
/* 244 */      NdrFcShort( 0x0 ),     /*
0 */

       /* Return value */

/* 246 */      NdrFcShort( 0x70 ),    /*
Flags:  out, return, base type, */
/* 248 */      NdrFcShort( 0x8 ),     /*
ia64, axp64 Stack size/offset = 8 */
```

```
/* 250 */      0x8,           /* FC_LONG
*/

                      0x0,           /*
0 */

                      0x0
        }
    };

static const MIDL_TYPE_FORMAT_STRING
__MIDL_TypeFormatString =
    {
        0,
        {
                      NdrFcShort( 0x0 ),
       /* 0 */
/*  2 */
                      0x12, 0x0,     /*
FC_UP */
/*  4 */       NdrFcShort( 0x39e ),   /*
Offset= 926 (930) */
/*  6 */
                      0x2b,          /*
FC_NON_ENCAPSULATED_UNION */
                      0x9,           /*
FC_ULONG */
/*  8 */       0x7,           /* Corr
desc: FC_USHORT */
                      0x0,           /*
*/
/* 10 */       NdrFcShort( 0xfff8 ),  /*
-8 */
/* 12 */       NdrFcShort( 0x1 ),     /*
Corr flags:  early, */
/* 14 */       NdrFcShort( 0x2 ),     /*
Offset= 2 (16) */
/* 16 */       NdrFcShort( 0x10 ),    /*
16 */
/* 18 */       NdrFcShort( 0x2b ),    /*
43 */
/* 20 */       NdrFcLong( 0x3 ),      /*
3 */
/* 24 */       NdrFcShort( 0x8008 ),  /*
Simple arm type: FC_LONG */
/* 26 */       NdrFcLong( 0x11 ),     /*
17 */
/* 30 */       NdrFcShort( 0x8001 ),  /*
Simple arm type: FC_BYTE */
/* 32 */       NdrFcLong( 0x2 ),      /*
2 */
/* 36 */       NdrFcShort( 0x8006 ),  /*
Simple arm type: FC_SHORT */
/* 38 */       NdrFcLong( 0x4 ),      /*
4 */
/* 42 */       NdrFcShort( 0x800a ),  /*
Simple arm type: FC_FLOAT */
/* 44 */       NdrFcLong( 0x5 ),      /*
5 */
/* 48 */       NdrFcShort( 0x800c ),  /*
Simple arm type: FC_DOUBLE */
/* 50 */       NdrFcLong( 0xb ),      /*
11 */
/* 54 */       NdrFcShort( 0x8006 ),  /*
Simple arm type: FC_SHORT */
/* 56 */       NdrFcLong( 0xa ),      /*
10 */
/* 60 */       NdrFcShort( 0x8008 ),  /*
Simple arm type: FC_LONG */
/* 62 */       NdrFcLong( 0x6 ),      /*
6 */
/* 66 */       NdrFcShort( 0xd6 ),    /*
Offset= 214 (280) */
/* 68 */       NdrFcLong( 0x7 ),      /*
7 */
/* 72 */       NdrFcShort( 0x800c ),  /*
Simple arm type: FC_DOUBLE */
/* 74 */       NdrFcLong( 0x8 ),      /*
8 */
/* 78 */       NdrFcShort( 0xd0 ),    /*
Offset= 208 (286) */
/* 80 */       NdrFcLong( 0xd ),      /*
13 */
/* 84 */       NdrFcShort( 0xe4 ),    /*
Offset= 228 (312) */
/* 86 */       NdrFcLong( 0x9 ),      /*
9 */
/* 90 */       NdrFcShort( 0xf0 ),    /*
Offset= 240 (330) */
/* 92 */       NdrFcLong( 0x2000 ),   /*
8192 */
/* 96 */       NdrFcShort( 0xfc ),    /*
Offset= 252 (348) */
/* 98 */       NdrFcLong( 0x24 ),     /*
36 */
/* 102 */      NdrFcShort( 0x2f4 ),   /*
Offset= 756 (858) */
/* 104 */      NdrFcLong( 0x4024 ),   /*
16420 */
/* 108 */      NdrFcShort( 0x2ee ),   /*
Offset= 750 (858) */
/* 110 */      NdrFcLong( 0x4011 ),   /*
16401 */
/* 114 */      NdrFcShort( 0x2ec ),   /*
Offset= 748 (862) */
/* 116 */      NdrFcLong( 0x4002 ),   /*
16386 */
/* 120 */      NdrFcShort( 0x2ea ),   /*
Offset= 746 (866) */
/* 122 */      NdrFcLong( 0x4003 ),   /*
16387 */
/* 126 */      NdrFcShort( 0x2e8 ),   /*
Offset= 744 (870) */
/* 128 */      NdrFcLong( 0x4004 ),   /*
16388 */
/* 132 */      NdrFcShort( 0x2e6 ),   /*
Offset= 742 (874) */
/* 134 */      NdrFcLong( 0x4005 ),   /*
16389 */
/* 138 */      NdrFcShort( 0x2e4 ),   /*
Offset= 740 (878) */
/* 140 */      NdrFcLong( 0x400b ),   /*
16395 */
/* 144 */      NdrFcShort( 0x2d2 ),   /*
Offset= 722 (866) */
/* 146 */      NdrFcLong( 0x400a ),   /*
16394 */
/* 150 */      NdrFcShort( 0x2d0 ),   /*
Offset= 720 (870) */
/* 152 */      NdrFcLong( 0x4006 ),   /*
16390 */
/* 156 */      NdrFcShort( 0x2d6 ),   /*
Offset= 726 (882) */
/* 158 */      NdrFcLong( 0x4007 ),   /*
16391 */
/* 162 */      NdrFcShort( 0x2cc ),   /*
Offset= 716 (878) */
/* 164 */      NdrFcLong( 0x4008 ),   /*
16392 */
```

```
/* 168 */	NdrFcShort( 0x2ce ),	/* Offset= 718 (886) */
/* 170 */	NdrFcLong( 0x400d ),	/* 16397 */
/* 174 */	NdrFcShort( 0x2cc ),	/* Offset= 716 (890) */
/* 176 */	NdrFcLong( 0x4009 ),	/* 16393 */
/* 180 */	NdrFcShort( 0x2ca ),	/* Offset= 714 (894) */
/* 182 */	NdrFcLong( 0x6000 ),	/* 24576 */

/* 186 */	NdrFcShort( 0x2c8 ),	/* Offset= 712 (898) */
/* 188 */	NdrFcLong( 0x400c ),	/* 16396 */
/* 192 */	NdrFcShort( 0x2c6 ),	/* Offset= 710 (902) */
/* 194 */	NdrFcLong( 0x10 ),	/* 16 */
/* 198 */	NdrFcShort( 0x8002 ),	/* Simple arm type: FC_CHAR */
/* 200 */	NdrFcLong( 0x12 ),	/* 18 */
/* 204 */	NdrFcShort( 0x8006 ),	/* Simple arm type: FC_SHORT */
/* 206 */	NdrFcLong( 0x13 ),	/* 19 */
/* 210 */	NdrFcShort( 0x8008 ),	/* Simple arm type: FC_LONG */
/* 212 */	NdrFcLong( 0x16 ),	/* 22 */
/* 216 */	NdrFcShort( 0x8008 ),	/* Simple arm type: FC_LONG */
/* 218 */	NdrFcLong( 0x17 ),	/* 23 */
/* 222 */	NdrFcShort( 0x8008 ),	/* Simple arm type: FC_LONG */
/* 224 */	NdrFcLong( 0xe ),	/* 14 */
/* 228 */	NdrFcShort( 0x2aa ),	/* Offset= 682 (910) */
/* 230 */	NdrFcLong( 0x400e ),	/* 16398 */
/* 234 */	NdrFcShort( 0x2b0 ),	/* Offset= 688 (922) */
/* 236 */	NdrFcLong( 0x4010 ),	/* 16400 */
/* 240 */	NdrFcShort( 0x2ae ),	/* Offset= 686 (926) */
/* 242 */	NdrFcLong( 0x4012 ),	/* 16402 */
/* 246 */	NdrFcShort( 0x26c ),	/* Offset= 620 (866) */
/* 248 */	NdrFcLong( 0x4013 ),	/* 16403 */
/* 252 */	NdrFcShort( 0x26a ),	/* Offset= 618 (870) */
/* 254 */	NdrFcLong( 0x4016 ),	/* 16406 */
/* 258 */	NdrFcShort( 0x264 ),	/* Offset= 612 (870) */
/* 260 */	NdrFcLong( 0x4017 ),	/* 16407 */
/* 264 */	NdrFcShort( 0x25e ),	/* Offset= 606 (870) */
/* 266 */	NdrFcLong( 0x0 ),	/* 0 */
/* 270 */	NdrFcShort( 0x0 ),	/* Offset= 0 (270) */
/* 272 */	NdrFcLong( 0x1 ),	/* 1 */
/* 276 */	NdrFcShort( 0x0 ),	/* Offset= 0 (276) */
/* 278 */	NdrFcShort( 0xffffffff ),	/* Offset= -1 (277) */
/* 280 */
			0x15,		/* FC_STRUCT */
			0x7,		/* 7 */
/* 282 */	NdrFcShort( 0x8 ),	/* 8 */
/* 284 */	0xb,		/* FC_HYPER */
			0x5b,		/* FC_END */
/* 286 */
			0x12, 0x0,	/* FC_UP */
/* 288 */	NdrFcShort( 0xe ),	/* Offset= 14 (302) */
/* 290 */
			0x1b,		/* FC_CARRAY */
			0x1,		/* 1 */
/* 292 */	NdrFcShort( 0x2 ),	/* 2 */
/* 294 */	0x9,		/* Corr desc: FC_ULONG */
			0x0,		/*  */
/* 296 */	NdrFcShort( 0xfffc ),	/* -4 */
/* 298 */	NdrFcShort( 0x1 ),	/* Corr flags:  early, */
/* 300 */	0x6,		/* FC_SHORT */
			0x5b,		/* FC_END */
/* 302 */
			0x17,		/* FC_CSTRUCT */
			0x3,		/* 3 */
/* 304 */	NdrFcShort( 0x8 ),	/* 8 */
/* 306 */	NdrFcShort( 0xfffffff0 ),	/* Offset= -16 (290) */
/* 308 */	0x8,		/* FC_LONG */
			0x8,		/* FC_LONG */
/* 310 */	0x5c,		/* FC_PAD */
			0x5b,		/* FC_END */
/* 312 */
			0x2f,		/* FC_IP */
			0x5a,		/* FC_CONSTANT_IID */
/* 314 */	NdrFcLong( 0x0 ),	/* 0 */
/* 318 */	NdrFcShort( 0x0 ),	/* 0 */
```

Appendix A:
Source Code

```
/* 320 */      NdrFcShort( 0x0 ),    /*
0 */
/* 322 */      0xc0,          /* 192 */
                       0x0,           /*
0 */
/* 324 */      0x0,           /* 0 */
                       0x0,           /*
0 */
/* 326 */      0x0,           /* 0 */
                       0x0,           /*
0 */
/* 328 */      0x0,           /* 0 */
                       0x46,          /*
70 */
/* 330 */
                       0x2f,          /*
FC_IP */

                       0x5a,          /*
FC_CONSTANT_IID */
/* 332 */      NdrFcLong( 0x20400 ),  /*
132096 */
/* 336 */      NdrFcShort( 0x0 ),    /*
0 */
/* 338 */      NdrFcShort( 0x0 ),    /*
0 */
/* 340 */      0xc0,          /* 192 */
                       0x0,           /*
0 */
/* 342 */      0x0,           /* 0 */
                       0x0,           /*
0 */
/* 344 */      0x0,           /* 0 */
                       0x0,           /*
0 */
/* 346 */      0x0,           /* 0 */
                       0x46,          /*
70 */
/* 348 */
                       0x12, 0x10,    /*
FC_UP [pointer_deref] */
/* 350 */      NdrFcShort( 0x2 ),    /*
Offset= 2 (352) */
/* 352 */
                       0x12, 0x0,     /*
FC_UP */
/* 354 */      NdrFcShort( 0x1e6 ),  /*
Offset= 486 (840) */
/* 356 */
                       0x2a,          /*
FC_ENCAPSULATED_UNION */
                       0x89,          /*
137 */
/* 358 */      NdrFcShort( 0x20 ),   /*
32 */
/* 360 */      NdrFcShort( 0xa ),    /*
10 */
/* 362 */      NdrFcLong( 0x8 ),     /*
8 */
/* 366 */      NdrFcShort( 0x50 ),   /*
Offset= 80 (446) */
/* 368 */      NdrFcLong( 0xd ),     /*
13 */
/* 372 */      NdrFcShort( 0x70 ),   /*
Offset= 112 (484) */
/* 374 */      NdrFcLong( 0x9 ),     /*
9 */
/* 378 */      NdrFcShort( 0x90 ),   /*
Offset= 144 (522) */
/* 380 */      NdrFcLong( 0xc ),     /*
12 */
/* 384 */      NdrFcShort( 0xb0 ),   /*
Offset= 176 (560) */
/* 386 */      NdrFcLong( 0x24 ),    /*
36 */
/* 390 */      NdrFcShort( 0x104 ),  /*
Offset= 260 (650) */
/* 392 */      NdrFcLong( 0x800d ),  /*
32781 */
/* 396 */      NdrFcShort( 0x120 ),  /*
Offset= 288 (684) */
/* 398 */      NdrFcLong( 0x10 ),    /*
16 */
/* 402 */      NdrFcShort( 0x13a ),  /*
Offset= 314 (716) */
/* 404 */      NdrFcLong( 0x2 ),     /*
2 */
/* 408 */      NdrFcShort( 0x150 ),  /*
Offset= 336 (744) */
/* 410 */      NdrFcLong( 0x3 ),     /*
3 */
/* 414 */      NdrFcShort( 0x166 ),  /*
Offset= 358 (772) */
/* 416 */      NdrFcLong( 0x14 ),    /*
20 */
/* 420 */      NdrFcShort( 0x17c ),  /*
Offset= 380 (800) */
/* 422 */      NdrFcShort( 0xffffffff ),
               /* Offset= -1 (421) */
/* 424 */
                       0x21,          /*
FC_BOGUS_ARRAY */
                       0x3,           /*
3 */
/* 426 */      NdrFcShort( 0x0 ),    /*
0 */
/* 428 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                       0x0,           /*
*/
/* 430 */      NdrFcShort( 0x0 ),    /*
0 */
/* 432 */      NdrFcShort( 0x1 ),    /*
Corr flags:  early, */
/* 434 */      NdrFcLong( 0xffffffff ),
               /* -1 */
/* 438 */      NdrFcShort( 0x0 ),    /*
Corr flags:  */
/* 440 */
                       0x12, 0x0,     /*
FC_UP */
/* 442 */      NdrFcShort( 0xffffff74 ),
               /* Offset= -140 (302) */
/* 444 */      0x5c,          /* FC_PAD
*/
                       0x5b,          /*
FC_END */
/* 446 */
                       0x1a,          /*
FC_BOGUS_STRUCT */
                       0x3,           /*
3 */
/* 448 */      NdrFcShort( 0x10 ),   /*
16 */
/* 450 */      NdrFcShort( 0x0 ),    /*
0 */
/* 452 */      NdrFcShort( 0x6 ),    /*
Offset= 6 (458) */
```

```
/* 454 */      0x8,           /* FC_LONG
*/
                         0x39,          /*
FC_ALIGNM8 */
/* 456 */      0x36,          /*
FC_POINTER */
                         0x5b,          /*
FC_END */
/* 458 */
                         0x11, 0x0,     /*
FC_RP */
/* 460 */      NdrFcShort( 0xffffffdc ),
        /* Offset= -36 (424) */
/* 462 */
                         0x21,          /*
FC_BOGUS_ARRAY */
                         0x3,           /*
3 */
/* 464 */      NdrFcShort( 0x0 ),     /*
0 */
/* 466 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                         0x0,           /*
*/
/* 468 */      NdrFcShort( 0x0 ),     /*
0 */
/* 470 */      NdrFcShort( 0x1 ),     /*
Corr flags:  early, */
/* 472 */      NdrFcLong( 0xffffffff ),
        /* -1 */
/* 476 */      NdrFcShort( 0x0 ),     /*
Corr flags:  */
/* 478 */      0x4c,          /*
FC_EMBEDDED_COMPLEX */
                         0x0,           /*
0 */
/* 480 */      NdrFcShort( 0xffffff58 ),
        /* Offset= -168 (312) */
/* 482 */      0x5c,          /* FC_PAD
*/
                         0x5b,          /*
FC_END */
/* 484 */
                         0x1a,          /*
FC_BOGUS_STRUCT */
                         0x3,           /*
3 */
/* 486 */      NdrFcShort( 0x10 ),    /*
16 */
/* 488 */      NdrFcShort( 0x0 ),     /*
0 */
/* 490 */      NdrFcShort( 0x6 ),     /*
Offset= 6 (496) */
/* 492 */      0x8,           /* FC_LONG
*/
                         0x39,          /*
FC_ALIGNM8 */
/* 494 */      0x36,          /*
FC_POINTER */
                         0x5b,          /*
FC_END */
/* 496 */
                         0x11, 0x0,     /*
FC_RP */
/* 498 */      NdrFcShort( 0xffffffdc ),
        /* Offset= -36 (462) */
/* 500 */
                         0x21,          /*
FC_BOGUS_ARRAY */
                         0x3,           /*
3 */
/* 502 */      NdrFcShort( 0x0 ),     /*
0 */
/* 504 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                         0x0,           /*
*/
/* 506 */      NdrFcShort( 0x0 ),     /*
0 */
/* 508 */      NdrFcShort( 0x1 ),     /*
Corr flags:  early, */
/* 510 */      NdrFcLong( 0xffffffff ),
        /* -1 */
/* 514 */      NdrFcShort( 0x0 ),     /*
Corr flags:  */
/* 516 */      0x4c,          /*
FC_EMBEDDED_COMPLEX */
                         0x0,           /*
0 */
/* 518 */      NdrFcShort( 0xffffff44 ),
        /* Offset= -188 (330) */
/* 520 */      0x5c,          /* FC_PAD
*/
                         0x5b,          /*
FC_END */
/* 522 */
                         0x1a,          /*
FC_BOGUS_STRUCT */
                         0x3,           /*
3 */
/* 524 */      NdrFcShort( 0x10 ),    /*
16 */
/* 526 */      NdrFcShort( 0x0 ),     /*
0 */
/* 528 */      NdrFcShort( 0x6 ),     /*
Offset= 6 (534) */
/* 530 */      0x8,           /* FC_LONG
*/
                         0x39,          /*
FC_ALIGNM8 */
/* 532 */      0x36,          /*
FC_POINTER */
                         0x5b,          /*
FC_END */
/* 534 */
                         0x11, 0x0,     /*
FC_RP */
/* 536 */      NdrFcShort( 0xffffffdc ),
        /* Offset= -36 (500) */
/* 538 */
                         0x21,          /*
FC_BOGUS_ARRAY */
                         0x3,           /*
3 */
/* 540 */      NdrFcShort( 0x0 ),     /*
0 */
/* 542 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                         0x0,           /*
*/
/* 544 */      NdrFcShort( 0x0 ),     /*
0 */
/* 546 */      NdrFcShort( 0x1 ),     /*
Corr flags:  early, */
/* 548 */      NdrFcLong( 0xffffffff ),
        /* -1 */
/* 552 */      NdrFcShort( 0x0 ),     /*
Corr flags:  */
/* 554 */
```

```
                        0x12, 0x0,      /*
FC_UP */
/* 556 */       NdrFcShort( 0x176 ),    /*
Offset= 374 (930) */
/* 558 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 560 */
                        0x1a,           /*
FC_BOGUS_STRUCT */
                        0x3,            /*
3 */
/* 562 */       NdrFcShort( 0x10 ),     /*
16 */
/* 564 */       NdrFcShort( 0x0 ),      /*
0 */
/* 566 */       NdrFcShort( 0x6 ),      /*
Offset= 6 (572) */
/* 568 */       0x8,            /* FC_LONG
*/
                        0x39,           /*
FC_ALIGNM8 */
/* 570 */       0x36,           /*
FC_POINTER */
                        0x5b,           /*
FC_END */
/* 572 */
                        0x11, 0x0,      /*
FC_RP */
/* 574 */       NdrFcShort( 0xffffffdc ),
        /* Offset= -36 (538) */
/* 576 */
                        0x2f,           /*
FC_IP */
                        0x5a,           /*
FC_CONSTANT_IID */
/* 578 */       NdrFcLong( 0x2f ),      /*
47 */
/* 582 */       NdrFcShort( 0x0 ),      /*
0 */
/* 584 */       NdrFcShort( 0x0 ),      /*
0 */
/* 586 */       0xc0,           /* 192 */
                        0x0,            /*
0 */
/* 588 */       0x0,            /* 0 */
                        0x0,            /*
0 */
/* 590 */       0x0,            /* 0 */
                        0x0,            /*
0 */
/* 592 */       0x0,            /* 0 */
                        0x46,           /*
70 */
/* 594 */
                        0x1b,           /*
FC_CARRAY */
                        0x0,            /*
0 */
/* 596 */       NdrFcShort( 0x1 ),      /*
1 */
/* 598 */       0x19,           /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,            /*
*/
/* 600 */       NdrFcShort( 0x4 ),      /*
4 */
/* 602 */       NdrFcShort( 0x1 ),      /*
Corr flags:  early, */
/* 604 */       0x1,            /* FC_BYTE
*/
                        0x5b,           /*
FC_END */
/* 606 */
                        0x1a,           /*
FC_BOGUS_STRUCT */
                        0x3,            /*
3 */
/* 608 */       NdrFcShort( 0x18 ),     /*
24 */
/* 610 */       NdrFcShort( 0x0 ),      /*
0 */
/* 612 */       NdrFcShort( 0xc ),      /*
Offset= 12 (624) */
/* 614 */       0x8,            /* FC_LONG
*/
                        0x8,            /*
FC_LONG */
/* 616 */       0x4c,           /*
FC_EMBEDDED_COMPLEX */
                        0x0,            /*
0 */
/* 618 */       NdrFcShort( 0xffffffd6 ),
        /* Offset= -42 (576) */
/* 620 */       0x39,           /*
FC_ALIGNM8 */
                        0x36,           /*
FC_POINTER */
/* 622 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 624 */
                        0x12, 0x0,      /*
FC_UP */
/* 626 */       NdrFcShort( 0xffffffe0 ),
        /* Offset= -32 (594) */
/* 628 */
                        0x21,           /*
FC_BOGUS_ARRAY */
                        0x3,            /*
3 */
/* 630 */       NdrFcShort( 0x0 ),      /*
0 */
/* 632 */       0x19,           /* Corr
desc:  field pointer, FC_ULONG */
                        0x0,            /*
*/
/* 634 */       NdrFcShort( 0x0 ),      /*
0 */
/* 636 */       NdrFcShort( 0x1 ),      /*
Corr flags:  early, */
/* 638 */       NdrFcLong( 0xffffffff ),
        /* -1 */
/* 642 */       NdrFcShort( 0x0 ),      /*
Corr flags:  */
/* 644 */
                        0x12, 0x0,      /*
FC_UP */
/* 646 */       NdrFcShort( 0xffffffd8 ),
        /* Offset= -40 (606) */
/* 648 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 650 */
                        0x1a,           /*
FC_BOGUS_STRUCT */
```

```
			0x3,		/* 
3 */
/* 652 */	NdrFcShort( 0x10 ),	/* 
16 */
/* 654 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 656 */	NdrFcShort( 0x6 ),	/* 
Offset= 6 (662) */
/* 658 */	0x8,		/* FC_LONG 
*/
			0x39,		/* 
FC_ALIGNM8 */
/* 660 */	0x36,		/* 
FC_POINTER */
			0x5b,		/* 
FC_END */
/* 662 */
			0x11, 0x0,	/* 
FC_RP */
/* 664 */	NdrFcShort( 0xffffffdc ),
	/* Offset= -36 (628) */
/* 666 */
			0x1d,		/* 
FC_SMFARRAY */
			0x0,		/* 
0 */
/* 668 */	NdrFcShort( 0x8 ),	/* 
8 */
/* 670 */	0x1,		/* FC_BYTE 
*/
			0x5b,		/* 
FC_END */
/* 672 */
			0x15,		/* 
FC_STRUCT */
			0x3,		/* 
3 */
/* 674 */	NdrFcShort( 0x10 ),	/* 
16 */
/* 676 */	0x8,		/* FC_LONG 
*/

			0x6,		/* 
FC_SHORT */
/* 678 */	0x6,		/* FC_SHORT 
*/
			0x4c,		/* 
FC_EMBEDDED_COMPLEX */
/* 680 */	0x0,		/* 0 */
			NdrFcShort( 
0xfffffff1 ),	/* Offset= -15 (666) */
			0x5b,		/* 
FC_END */
/* 684 */
			0x1a,		/* 
FC_BOGUS_STRUCT */
			0x3,		/* 
3 */
/* 686 */	NdrFcShort( 0x20 ),	/* 
32 */
/* 688 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 690 */	NdrFcShort( 0xa ),	/* 
Offset= 10 (700) */
/* 692 */	0x8,		/* FC_LONG 
*/
			0x39,		/* 
FC_ALIGNM8 */
/* 694 */	0x36,		/* 
FC_POINTER */
			0x4c,		/* 
FC_EMBEDDED_COMPLEX */
/* 696 */	0x0,		/* 0 */
			NdrFcShort( 
0xffffffe7 ),	/* Offset= -25 (672) */
			0x5b,		/* 
FC_END */
/* 700 */
			0x11, 0x0,	/* 
FC_RP */
/* 702 */	NdrFcShort( 0xffffff10 ),
	/* Offset= -240 (462) */
/* 704 */
			0x1b,		/* 
FC_CARRAY */
			0x0,		/* 
0 */
/* 706 */	NdrFcShort( 0x1 ),	/* 
1 */
/* 708 */	0x19,		/* Corr 
desc:  field pointer, FC_ULONG */
			0x0,		/* 
*/
/* 710 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 712 */	NdrFcShort( 0x1 ),	/* 
Corr flags:  early, */
/* 714 */	0x1,		/* FC_BYTE 
*/
			0x5b,		/* 
FC_END */
/* 716 */
			0x1a,		/* 
FC_BOGUS_STRUCT */
			0x3,		/* 
3 */
/* 718 */	NdrFcShort( 0x10 ),	/* 
16 */
/* 720 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 722 */	NdrFcShort( 0x6 ),	/* 
Offset= 6 (728) */
/* 724 */	0x8,		/* FC_LONG 
*/
			0x39,		/* 
FC_ALIGNM8 */
/* 726 */	0x36,		/* 
FC_POINTER */
			0x5b,		/* 
FC_END */
/* 728 */
			0x12, 0x0,	/* 
FC_UP */
/* 730 */	NdrFcShort( 0xffffffe6 ),
	/* Offset= -26 (704) */
/* 732 */
			0x1b,		/* 
FC_CARRAY */
			0x1,		/* 
1 */
/* 734 */	NdrFcShort( 0x2 ),	/* 
2 */
/* 736 */	0x19,		/* Corr 
desc:  field pointer, FC_ULONG */
			0x0,		/* 
*/
/* 738 */	NdrFcShort( 0x0 ),	/* 
0 */
/* 740 */	NdrFcShort( 0x1 ),	/* 
Corr flags:  early, */
```

Appendix A: Source Code

```
/* 742 */      0x6,           /* FC_SHORT
*/
                      0x5b,          /*
FC_END */
/* 744 */

                      0x1a,          /*
FC_BOGUS_STRUCT */
                      0x3,           /*
3 */
/* 746 */      NdrFcShort( 0x10 ),   /*
16 */
/* 748 */      NdrFcShort( 0x0 ),    /*
0 */
/* 750 */      NdrFcShort( 0x6 ),    /*
Offset= 6 (756) */
/* 752 */      0x8,           /* FC_LONG
*/
                      0x39,          /*
FC_ALIGNM8 */
/* 754 */      0x36,          /*
FC_POINTER */
                      0x5b,          /*
FC_END */
/* 756 */
                      0x12, 0x0,     /*
FC_UP */
/* 758 */      NdrFcShort( 0xffffffe6 ),
        /* Offset= -26 (732) */
/* 760 */
                      0x1b,          /*
FC_CARRAY */
                      0x3,           /*
3 */
/* 762 */      NdrFcShort( 0x4 ),    /*
4 */
/* 764 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                      0x0,           /*
*/
/* 766 */      NdrFcShort( 0x0 ),    /*
0 */
/* 768 */      NdrFcShort( 0x1 ),    /*
Corr flags:  early, */
/* 770 */      0x8,           /* FC_LONG
*/
                      0x5b,          /*
FC_END */
/* 772 */
                      0x1a,          /*
FC_BOGUS_STRUCT */
                      0x3,           /*
3 */
/* 774 */      NdrFcShort( 0x10 ),   /*
16 */
/* 776 */      NdrFcShort( 0x0 ),    /*
0 */
/* 778 */      NdrFcShort( 0x6 ),    /*
Offset= 6 (784) */
/* 780 */      0x8,           /* FC_LONG
*/
                      0x39,          /*
FC_ALIGNM8 */
/* 782 */      0x36,          /*
FC_POINTER */
                      0x5b,          /*
FC_END */
/* 784 */
                      0x12, 0x0,     /*
FC_UP */
/* 786 */      NdrFcShort( 0xffffffe6 ),
        /* Offset= -26 (760) */
/* 788 */
                      0x1b,          /*
FC_CARRAY */
                      0x7,           /*
7 */
/* 790 */      NdrFcShort( 0x8 ),    /*
8 */
/* 792 */      0x19,          /* Corr
desc:  field pointer, FC_ULONG */
                      0x0,           /*
*/
/* 794 */      NdrFcShort( 0x0 ),    /*
0 */
/* 796 */      NdrFcShort( 0x1 ),    /*
Corr flags:  early, */
/* 798 */      0xb,           /* FC_HYPER
*/
                      0x5b,          /*
FC_END */
/* 800 */
                      0x1a,          /*
FC_BOGUS_STRUCT */
                      0x3,           /*
3 */
/* 802 */      NdrFcShort( 0x10 ),   /*
16 */
/* 804 */      NdrFcShort( 0x0 ),    /*
0 */
/* 806 */      NdrFcShort( 0x6 ),    /*
Offset= 6 (812) */
/* 808 */      0x8,           /* FC_LONG
*/
                      0x39,          /*
FC_ALIGNM8 */
/* 810 */      0x36,          /*
FC_POINTER */
                      0x5b,          /*
FC_END */
/* 812 */
                      0x12, 0x0,     /*
FC_UP */
/* 814 */      NdrFcShort( 0xffffffe6 ),
        /* Offset= -26 (788) */
/* 816 */
                      0x15,          /*
FC_STRUCT */
                      0x3,           /*
3 */
/* 818 */      NdrFcShort( 0x8 ),    /*
8 */
/* 820 */      0x8,           /* FC_LONG
*/
                      0x8,           /*
FC_LONG */
/* 822 */      0x5c,          /* FC_PAD
*/
                      0x5b,          /*
FC_END */
/* 824 */
                      0x1b,          /*
FC_CARRAY */
                      0x3,           /*
3 */
/* 826 */      NdrFcShort( 0x8 ),    /*
8 */
/* 828 */      0x7,           /* Corr
desc: FC_USHORT */
```

```
                        0x0,            /*
*/
/* 830 */       NdrFcShort( 0xffc8 ),  /*
-56 */
/* 832 */       NdrFcShort( 0x1 ),     /*
Corr flags:  early, */
/* 834 */       0x4c,           /*
FC_EMBEDDED_COMPLEX */
                        0x0,            /*
0 */
/* 836 */       NdrFcShort( 0xffffffec ),
        /* Offset= -20 (816) */
/* 838 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 840 */
                        0x1a,           /*
FC_BOGUS_STRUCT */
                        0x3,            /*
3 */
/* 842 */       NdrFcShort( 0x38 ),    /*
56 */
/* 844 */       NdrFcShort( 0xffffffec ),
        /* Offset= -20 (824) */
/* 846 */       NdrFcShort( 0x0 ),     /*
Offset= 0 (846) */
/* 848 */       0x6,            /* FC_SHORT
*/
                        0x6,            /*
FC_SHORT */
/* 850 */       0x38,           /*
FC_ALIGNM4 */
                        0x8,            /*
FC_LONG */
/* 852 */       0x8,            /* FC_LONG
*/
                        0x4c,           /*
FC_EMBEDDED_COMPLEX */
/* 854 */       0x4,            /* 4 */
                        NdrFcShort(
0xfffffe0d ),  /* Offset= -499 (356) */
                        0x5b,           /*
FC_END */
/* 858 */
                        0x12, 0x0,      /*
FC_UP */
/* 860 */       NdrFcShort( 0xffffff02 ),
        /* Offset= -254 (606) */
/* 862 */
                        0x12, 0x8,      /*
FC_UP [simple_pointer] */
/* 864 */       0x1,            /* FC_BYTE
*/
                        0x5c,           /*
FC_PAD */
/* 866 */
                        0x12, 0x8,      /*
FC_UP [simple_pointer] */
/* 868 */       0x6,            /* FC_SHORT
*/
                        0x5c,           /*
FC_PAD */
/* 870 */
                        0x12, 0x8,      /*
FC_UP [simple_pointer] */
/* 872 */       0x8,            /* FC_LONG
*/
                        0x5c,           /*
FC_PAD */
/* 874 */
                        0x12, 0x8,      /*
FC_UP [simple_pointer] */
/* 876 */       0xa,            /* FC_FLOAT
*/
                        0x5c,           /*
FC_PAD */
/* 878 */
                        0x12, 0x8,      /*
FC_UP [simple_pointer] */
/* 880 */       0xc,            /*
FC_DOUBLE */
                        0x5c,           /*
FC_PAD */
/* 882 */
                        0x12, 0x0,      /*
FC_UP */
/* 884 */       NdrFcShort( 0xfffffda4 ),
        /* Offset= -604 (280) */
/* 886 */
                        0x12, 0x10,     /*
FC_UP [pointer_deref] */
/* 888 */       NdrFcShort( 0xfffffda6 ),
        /* Offset= -602 (286) */
/* 890 */
                        0x12, 0x10,     /*
FC_UP [pointer_deref] */
/* 892 */       NdrFcShort( 0xfffffdbc ),
        /* Offset= -580 (312) */
/* 894 */
                        0x12, 0x10,     /*
FC_UP [pointer_deref] */
/* 896 */       NdrFcShort( 0xfffffdca ),
        /* Offset= -566 (330) */
/* 898 */
                        0x12, 0x10,     /*
FC_UP [pointer_deref] */
/* 900 */       NdrFcShort( 0xfffffdd8 ),
        /* Offset= -552 (348) */
/* 902 */
                        0x12, 0x10,     /*
FC_UP [pointer_deref] */
/* 904 */       NdrFcShort( 0x2 ),     /*
Offset= 2 (906) */
/* 906 */
                        0x12, 0x0,      /*
FC_UP */
/* 908 */       NdrFcShort( 0x16 ),    /*
Offset= 22 (930) */
/* 910 */
                        0x15,           /*
FC_STRUCT */
                        0x7,            /*
7 */
/* 912 */       NdrFcShort( 0x10 ),    /*
16 */
/* 914 */       0x6,            /* FC_SHORT
*/
                        0x1,            /*
FC_BYTE */
/* 916 */       0x1,            /* FC_BYTE
*/
                        0x38,           /*
FC_ALIGNM4 */
/* 918 */       0x8,            /* FC_LONG
*/
                        0x39,           /*
FC_ALIGNM8 */
/* 920 */       0xb,            /* FC_HYPER
*/
```

```
                        0x5b,           /*
FC_END */
/* 922 */
                        0x12, 0x0,      /*
FC_UP */
/* 924 */       NdrFcShort( 0xfffffff2 ),
        /* Offset= -14 (910) */
/* 926 */
                        0x12, 0x8,      /*
FC_UP [simple_pointer] */
/* 928 */       0x2,            /* FC_CHAR
*/
                        0x5c,           /*
FC_PAD */
/* 930 */
                        0x1a,           /*
FC_BOGUS_STRUCT */
                        0x7,            /*
7 */
/* 932 */       NdrFcShort( 0x20 ),     /*
32 */
/* 934 */       NdrFcShort( 0x0 ),      /*
0 */
/* 936 */       NdrFcShort( 0x0 ),      /*
Offset= 0 (936) */
/* 938 */       0x8,            /* FC_LONG
*/
                        0x8,            /*
FC_LONG */
/* 940 */       0x6,            /* FC_SHORT
*/
                        0x6,            /*
FC_SHORT */
/* 942 */       0x6,            /* FC_SHORT
*/
                        0x6,            /*
FC_SHORT */
/* 944 */       0x4c,           /*
FC_EMBEDDED_COMPLEX */
                        0x0,            /*
0 */
/* 946 */       NdrFcShort( 0xfffffc54 ),
        /* Offset= -940 (6) */
/* 948 */       0x5c,           /* FC_PAD
*/
                        0x5b,           /*
FC_END */
/* 950 */       0xb4,           /*
FC_USER_MARSHAL */
                        0x83,           /*
131 */
/* 952 */       NdrFcShort( 0x0 ),      /*
0 */
/* 954 */       NdrFcShort( 0x18 ),     /*
24 */
/* 956 */       NdrFcShort( 0x0 ),      /*
0 */
/* 958 */       NdrFcShort( 0xfffffc44 ),
        /* Offset= -956 (2) */
/* 960 */
                        0x11, 0x4,      /*
FC_RP [alloced_on_stack] */
/* 962 */       NdrFcShort( 0x6 ),      /*
Offset= 6 (968) */
/* 964 */
                        0x13, 0x0,      /*
FC_OP */
/* 966 */       NdrFcShort( 0xffffffdc ),
        /* Offset= -36 (930) */
/* 968 */       0xb4,           /*
FC_USER_MARSHAL */
                        0x83,           /*
131 */
/* 970 */       NdrFcShort( 0x0 ),      /*
0 */
/* 972 */       NdrFcShort( 0x18 ),     /*
24 */
/* 974 */       NdrFcShort( 0x0 ),      /*
0 */
/* 976 */       NdrFcShort( 0xfffffff4 ),
        /* Offset= -12 (964) */

                        0x0
        }
    };

const CInterfaceProxyVtbl *
_tpcc_com_ps_ProxyVtblList[] =
{
    ( CInterfaceProxyVtbl *)
&_ITPCCProxyVtbl,
    0
};

const CInterfaceStubVtbl *
_tpcc_com_ps_StubVtblList[] =
{
    ( CInterfaceStubVtbl *)
&_ITPCCStubVtbl,
    0
};

PCInterfaceName const
_tpcc_com_ps_InterfaceNamesList[] =
{
    "ITPCC",
    0
};


#define _tpcc_com_ps_CHECK_IID(n)
        IID_GENERIC_CHECK_IID(
_tpcc_com_ps, pIID, n)

int __stdcall _tpcc_com_ps_IID_Lookup(
const IID * pIID, int * pIndex )
{

    if(!_tpcc_com_ps_CHECK_IID(0))
        {
        *pIndex = 0;
        return 1;
        }

    return 0;
}

const ExtendedProxyFileInfo
tpcc_com_ps_ProxyFileInfo =
{
    (PCInterfaceProxyVtblList *) &
_tpcc_com_ps_ProxyVtblList,
    (PCInterfaceStubVtblList *) &
_tpcc_com_ps_StubVtblList,
    (const PCInterfaceName * ) &
_tpcc_com_ps_InterfaceNamesList,
    0, // no delegation
    & _tpcc_com_ps_IID_Lookup,
    1,
```

```
    2,
    0, /* table of [async_uuid]
interfaces */
    0, /* Filler1 */
    0, /* Filler2 */
    0  /* Filler3 */
};


#endif /* defined(_M_IA64) ||
defined(_M_AXP64)*/
```

Appendix B:
Database Design

CreateDB.sql

```
--  File:     CREATEDB.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates tpcc database and
backup files


use master
go

--      Create temporary table for timing

if exists ( select name from sysobjects
where name = 'tpcc_timer' )
        drop table tpcc_timer
go

create table tpcc_timer
(
        start_date
        char(30),
        end_date
        char(30)
)

insert  into tpcc_timer values (0,0)
go

--      Store starting time

update  tpcc_timer
set     start_date      = (select
convert(char(30), getdate(),9))
go

--  create main database files

CREATE DATABASE tpcc
ON PRIMARY
(       NAME            = MSSQL_tpcc_root,
        FILENAME        =
"C:\MSSQL_tpcc_root.mdf",
        SIZE            = 8MB,
        FILEGROWTH      = 0),
FILEGROUP      MSSQL_misc_fg
(       NAME            = MSSQL_misc1,
        FILENAME        = "F:",
        SIZE            = 43620MB,
        FILEGROWTH      = 0),
(       NAME            = MSSQL_misc2,
        FILENAME        = "G:",
        SIZE            = 56460MB,
        FILEGROWTH      = 0),
(       NAME            = MSSQL_misc3,
        FILENAME        = "H:",
        SIZE            = 56460MB,
        FILEGROWTH      = 0),
(       NAME            = MSSQL_misc4,
        FILENAME        = "I:",
        SIZE            = 43620MB,
        FILEGROWTH      = 0),
(       NAME            = MSSQL_misc5,
        FILENAME        = "J:",
        SIZE            = 30780MB,
        FILEGROWTH      = 0),
FILEGROUP      MSSQL_cs_fg
(       NAME            = MSSQL_cs1,
        FILENAME        = "M:",
        SIZE            = 80340MB,
        FILEGROWTH      = 0),
(       NAME            = MSSQL_cs2,
        FILENAME        = "N:",
        SIZE            = 103980MB,
        FILEGROWTH      = 0),
(       NAME            = MSSQL_cs3,
        FILENAME        = "O:",
        SIZE            = 103980MB,
        FILEGROWTH      = 0),
(       NAME            = MSSQL_cs4,
        FILENAME        = "P:",
        SIZE            = 80340MB,
        FILEGROWTH      = 0),
(       NAME            = MSSQL_cs5,
        FILENAME        = "Q:",
        SIZE            = 56700MB,
        FILEGROWTH      = 0)
LOG ON
(       NAME            = MSSQL_tpcc_log,
        FILENAME        = "E:",
        SIZE            = 204720MB,
        FILEGROWTH      = 0)
COLLATE Latin1_General_BIN
go

-- Store ending time
update  tpcc_timer
set     end_date        = (select
convert(char(30), getdate(),9))
go

select "Elapsed time (in seconds): ",
datediff(second,(select start_date from
tpcc_timer),(select end_date from
tpcc_timer))

--      remove temporary table

if exists ( select name from sysobjects
where name = 'tpcc_timer' )
        drop table tpcc_timer
go
```

BackupDev.sql

```
--  File:     BACKUPDEVB.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates tpcc database
Backup Devices


use master
go
```

```
--  create backup devices

exec sp_addumpdevice
'disk','tpccback1','T:\tpccback1.dmp'
go
exec sp_addumpdevice
'disk','tpccback2','U:\tpccback2.dmp'
go
exec sp_addumpdevice
'disk','tpccback3','V:\tpccback3.dmp'
go
exec sp_addumpdevice
'disk','tpccback4','V:\tpccback4.dmp'
go
```

Backup.sql

```
--  File:     BACKUP.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates backup of tpcc
database


declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

dump database tpcc to tpccback1,
tpccback2, tpccback3, tpccback4 with
init, stats = 1

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Tables.sql

```
--  File:     TABLES.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates TPC-C tables


use tpcc
go

--
--  Remove all existing TPC-C tables
--

if exists ( select name from sysobjects
where name = 'warehouse' )
       drop table warehouse
go
if exists ( select name from sysobjects
where name = 'district' )
       drop table district
go
if exists ( select name from sysobjects
where name = 'customer' )
       drop table customer
go
if exists ( select name from sysobjects
where name = 'history' )
       drop table history
go
if exists ( select name from sysobjects
where name = 'new_order' )
       drop table new_order
go
if exists ( select name from sysobjects
where name = 'orders' )
       drop table orders
go
if exists ( select name from sysobjects
where name = 'order_line' )
       drop table order_line
go
if exists ( select name from sysobjects
where name = 'item' )
       drop table item
go
if exists ( select name from sysobjects
where name = 'stock' )
       drop table stock
go

--
--  Create new tables
--

create table warehouse
(
       w_id
       smallint,
       w_name
       char(10),
       w_street_1
       char(20),
       w_street_2
       char(20),
       w_city
       char(20),
       w_state
       char(2),
       w_zip
       char(9),
       w_tax
       numeric(4,4),
       w_ytd
       numeric(12,2)
) on MSSQL_misc_fg
go

create table district
(
       d_id
       tinyint,
       d_w_id
       smallint,
       d_name
       char(10),
       d_street_1
       char(20),
       d_street_2
       char(20),
```

```
	d_city
	char(20),
	d_state
	char(2),
	d_zip
	char(9),
	d_tax
	numeric(4,4),
	d_ytd
	numeric(12,2),
	d_next_o_id			int
) on MSSQL_misc_fg
go

create table customer
(
	c_id
	int,
	c_d_id
	tinyint,
	c_w_id
	smallint,
	c_first
	char(16),
	c_middle
	char(2),
	c_last
	char(16),
	c_street_1
	char(20),
	c_street_2
	char(20),
	c_city
	char(20),
	c_state
	char(2),
	c_zip
	char(9),
	c_phone
	char(16),
	c_since
	datetime,
	c_credit
	char(2),
	c_credit_lim
	numeric(12,2),
	c_discount
	numeric(4,4),
	c_balance
	numeric(12,2),
	c_ytd_payment
	numeric(12,2),
	c_payment_cnt		smallint,
	c_delivery_cnt		smallint,
	c_data
	char(500)
) on MSSQL_cs_fg
go

create table history
(
	h_c_id
	int,
	h_c_d_id
	tinyint,
	h_c_w_id
	smallint,
	h_d_id
	tinyint,
	h_w_id
	smallint,
	h_date
	datetime,
	h_amount
	numeric(6,2),
	h_data
	char(24)
) on MSSQL_misc_fg
go

create table new_order
(
	no_o_id
	int,
	no_d_id
	tinyint,
	no_w_id
	smallint
) on MSSQL_misc_fg
go

create table orders
(
	o_id
	int,
	o_d_id
	tinyint,
	o_w_id
	smallint,
	o_c_id
	int,
	o_entry_d
	datetime,
	o_carrier_id		tinyint,
	o_ol_cnt
	tinyint,
	o_all_local
	tinyint
) on MSSQL_misc_fg
go

create table order_line
(
	ol_o_id
	int,
	ol_d_id
	tinyint,
	ol_w_id
	smallint,
	ol_number
	tinyint,
	ol_i_id
	int,
	ol_supply_w_id		smallint,
	ol_delivery_d		datetime,
	ol_quantity
	smallint,
	ol_amount
	numeric(6,2),
	ol_dist_info		char(24)
) on MSSQL_misc_fg
go

create table item

(
	i_id
	int,
	i_im_id
	int,
	i_name
	char(24),
```

```
        i_price
        numeric(5,2),
        i_data
        char(50)
) on MSSQL_misc_fg
go

create table stock
(
        s_i_id
        int,
        s_w_id
        smallint,
        s_quantity
        smallint,
        s_dist_01
        char(24),
        s_dist_02
        char(24),
        s_dist_03
        char(24),
        s_dist_04
        char(24),
        s_dist_05
        char(24),
        s_dist_06
        char(24),
        s_dist_07
        char(24),
        s_dist_08
        char(24),
        s_dist_09
        char(24),
        s_dist_10
        char(24),
        s_ytd
        int,
        s_order_cnt
        smallint,
        s_remote_cnt          smallint,
        s_data
        char(50)
) on MSSQL_cs_fg
go
```

Idxcusnc.sql

```
--  File:     IDXCUSNC.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates non-clustered index
on customer table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where  name = 'customer_nc1' )
        drop index customer.customer_nc1

create unique nonclustered index
customer_nc1 on customer(c_w_id, c_d_id,
c_last, c_first, c_id)
        on MSSQL_cs_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxdiscl.sql

```
--  File:     IDXDISCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
district table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where name = 'district_c1' )
        drop index district.district_c1

create unique clustered index
district_c1 on district(d_w_id, d_id)
        with fillfactor=100 on
MSSQL_misc_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxhiscl.sql

```
--  File:     IDXHISCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
history table
--
--  CAUTION:
****************************************
--  CAUTION:  This index is only
beneficial for systems
--  CAUTION:  with 8 or more processors.
```

```
--  CAUTION:  It may negatively impact
performance on
--  CAUTION:  on systems with less than 8
processors.
--  CAUTION:
******************************************



use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where  name = 'history_c1' )
       drop index history.history_c1

create unique clustered index history_c1
on history(h_c_w_id, h_date, h_c_d_id,
h_c_id, h_amount)
       on MSSQL_tpcc_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxitmcl.sql

```
--  File:     IDXITMCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
item table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where  name = 'item_c1' )
       drop index item.item_c1

create unique clustered index item_c1 on
item(i_id)
       on MSSQL_misc_fg


select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
--  File:     IDXNODCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
new_order table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where  name = 'new_order_c1' )
       drop index new_order.new_order_c1

create unique clustered index
new_order_c1 on new_order(no_w_id,
no_d_id, no_o_id)
       on MSSQL_misc_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxodlcl.sql

```
--  File:     IDXODLCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
order_line table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where  name = 'order_line_c1' )
       drop index
order_line.order_line_c1

create unique clustered index
order_line_c1 on order_line(ol_w_id,
ol_d_id, ol_o_id, ol_number)
       on MSSQL_misc_fg

select @enddate = getdate()
```

```
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxordcl.sql

```
--  File:     IDXORDCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
orders table


use tpcc
go

declare @startdate datetime
declare @enddate datetime

select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where  name = 'orders_c1' )
       drop index orders.orders_c1

create unique clustered index orders_c1
on orders(o_w_id, o_d_id, o_id)
       on MSSQL_misc_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxordnc.sql

```
--  File:     IDXORDNC.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates non-clustered index
on orders table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where  name = 'orders_nc1' )
       drop index orders.orders_nc1

create index orders_nc1 on orders(o_w_id,
o_d_id, o_c_id, o_id)
       on MSSQL_misc_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxstkcl.sql

```
--  File:     IDXSTKCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
stock table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where  name = 'stock_c1' )
       drop index stock.stock_c1

create unique clustered index stock_c1 on
stock(s_i_id, s_w_id)
       on MSSQL_cs_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxwarcl.sql

```
--  File:     IDXWARCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
warehouse table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
```

```
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where name = 'warehouse_c1' )
       drop index warehouse.warehouse_c1

create unique clustered index
warehouse_c1 on warehouse(w_id)
       with fillfactor=100 on
MSSQL_misc_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

Idxcuscl.sql

```
--  File:     IDXCUSCL.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates clustered index on
customer table


use tpcc
go

declare @startdate datetime
declare @enddate datetime
select @startdate = getdate()
select "Start date:",
convert(varchar(30),@startdate,9)

if exists ( select name from sysindexes
where name = 'customer_c1' )
       drop index customer.customer_c1

create unique clustered index customer_c1
on customer(c_w_id, c_d_id, c_id)
       on MSSQL_cs_fg

select @enddate = getdate()
select "End date: ",
convert(varchar(30),@enddate,9)
select "Elapsed time (in seconds): ",
datediff(second, @startdate, @enddate)

go
```

VerifyTpccLoad.sql

```
--  File:     VERIFYTPCCLOAD.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Performs series of TPCC
database checks to verify
--            that database load
completed correctly

print  " "

select  convert(char(30), getdate(),9)
print  " "

use tpcc
go

--
       ********************************
***
--
--     Check rows per table from
SYSINDEXES
--
--
       ********************************
***

print  'WAREHOUSE TABLE'

select  rows
from    sysindexes
where   id     = object_id("warehouse")
go

print  'DISTRICT TABLE = (10 * No of
warehouses)'

select  rows
from    sysindexes
where   id     =object_id("district")
go

print  'ITEM TABLE = 100,000'

select  rows
from    sysindexes
where   id     =object_id("item")
go

print  'CUSTOMER TABLE = (30,000 * No of
warehouses)'

select  rows
from    sysindexes
where   id     =object_id("customer")
go

print 'ORDERS TABLE = (30,000 * No of
warehouses)'

select  rows
from    sysindexes
where   id     =object_id("orders")
go

print  'HISTORY TABLE = (30,000 * No of
warehouses)'

select  rows
from    sysindexes
where   id     =object_id("history")
go

print  'STOCK TABLE = (100,000 * No of
warehouses)'
```

```
select  rows
from    sysindexes
where   id      =object_id("stock")
go

print   'ORDER_LINE TABLE = (300,000 * No
of warehouses + some change)'

select  rows
from    sysindexes
where   id      =object_id("order_line")
go

print   'NEW_ORDER TABLE = (9000 * No of
warehouses)'

select  rows
from    sysindexes
where   id      =object_id("new_order")
go

--      *************
--
--      Check indices
--
--      *************

print '**************Index
Check****************'

use tpcc
go

sp_helpindex   customer
go

sp_helpindex   stock
go

sp_helpindex   district
go

sp_helpindex   item
go

sp_helpindex   new_order
go

sp_helpindex   orders
go

sp_helpindex   order_line
go

sp_helpindex   warehouse
go
```

Dbopt2.sql

```
--  File:      DBOPT2.SQL
--             Microsoft TPC-C Benchmark
Kit Ver. 4.41
--             Copyright Microsoft, 2001
--  Purpose:  Resets database options
after data load

exec sp_dboption tpcc,'select
into/bulkcopy',false
exec sp_dboption tpcc,'trunc. log on
chkpt.',false
exec sp_dboption tpcc,'torn page
detection',false
GO

USE tpcc
GO

CHECKPOINT
GO

sp_configure 'allow updates',1
GO

RECONFIGURE WITH OVERRIDE
GO

DECLARE @msg          varchar(50)

--
--
--        OPTIONS FOR SQL SERVER 2000
--
-- Set option values for user-defined
indexes --
--
--

SET     @msg    = ' '
PRINT   @msg
SET     @msg    = 'Setting SQL Server
indexoptions'
PRINT   @msg
SET     @msg    = ' '
PRINT   @msg

EXEC sp_indexoption    'customer',
        'DisAllowPageLocks',   TRUE
EXEC sp_indexoption    'district',
        'DisAllowPageLocks',   TRUE
EXEC sp_indexoption    'warehouse',
        'DisAllowPageLocks',   TRUE
EXEC sp_indexoption    'stock',
        'DisAllowPageLocks',   TRUE
EXEC sp_indexoption    'order_line',
        'DisAllowRowLocks',    TRUE
EXEC sp_indexoption    'orders',
        'DisAllowRowLocks',    TRUE
EXEC sp_indexoption    'new_order',
        'DisAllowRowLocks',    TRUE
EXEC sp_indexoption    'item',
        'DisAllowRowLocks',    TRUE
EXEC sp_indexoption    'item',
        'DisAllowPageLocks',   TRUE
GO

Print ' '
Print '******************'
Print 'Pre-specified Locking Hierarchy:'
Print '  Lockflag = 0 ==> No pre-
specified hierarchy'
Print '   Lockflag = 1 ==> Lock at Page-
level then Table-level'
Print '   Lockflag = 2 ==> Lock at Row-
level then Table-level'
Print '   Lockflag = 3 ==> Lock at Table-
level'
Print ' '

SELECT  name,lockflags
```

```
FROM    sysindexes
WHERE   object_id('warehouse')= id OR
        object_id('district')  = id OR
        object_id('customer')  = id OR
        object_id('stock')     = id OR
        object_id('orders')    = id OR
        object_id('order_line')       =
id OR
        object_id('history')   = id OR
        object_id('new_order')= id OR
        object_id('item')      = id
ORDER   BY lockflags asc
GO

sp_configure 'allow updates',0
GO

RECONFIGURE WITH OVERRIDE
GO

EXEC sp_dboption tpcc,'auto update
statistics',   FALSE
EXEC sp_dboption tpcc,'auto create
statistics',   FALSE
GO

EXEC sp_tableoption   'district',
        'pintable',true
EXEC sp_tableoption   'warehouse',
        'pintable',true
EXEC sp_tableoption   'new_order',
        'pintable',true
EXEC sp_tableoption   'item',
        'pintable',true
GO
```

Dbopt1.sql

```
--  File:     DBOPT1.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Sets database options for
data load


use master
go

exec sp_dboption tpcc,'select
into/bulkcopy',true
exec sp_dboption tpcc,'trunc. log on
chkpt.',true
exec sp_dboption tpcc,'torn page
detection',false
go

use tpcc
go

checkpoint
go
```

Stocklev.sql

```
--  File:     STOCKLEV.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates stock level
transaction stored procedure
--
--      Interface Level: 4.10.000

use tpcc
go

if exists (select name from sysobjects
where name = 'tpcc_stocklevel' )
        drop procedure tpcc_stocklevel
go

create proc tpcc_stocklevel   @w_id
        smallint,
                              @d_id
        tinyint,
                              @threshhold
        smallint
as

declare @o_id_low int,
        @o_id_high int

select  @o_id_low      = (d_next_o_id -
20),
        @o_id_high     = (d_next_o_id -
1)
from    district
where   d_w_id         = @w_id and
        d_id           = @d_id

select  count(distinct(s_i_id))
from    stock, order_line
where   ol_w_id        =  @w_id and
        ol_d_id        =  @d_id and
        ol_o_id           between
@o_id_low and
                          @o_id_high and
        s_w_id         =  ol_w_id and
        s_i_id         =  ol_i_id and
        s_quantity     <  @threshhold

go
```

Neword.sql

```
--  File:     NEWORD.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates new order
transaction stored procedure
--
--      Interface Level: 4.10.000

use tpcc
go
```

```
if exists ( select name from sysobjects
where name = 'tpcc_neworder' )
       drop procedure tpcc_neworder
go

create proc tpcc_neworder

       @w_id           smallint,

       @d_id           tinyint,

       @c_id           int,

       @o_ol_cnt       tinyint,

       @o_all_local    tinyint,

       @i_id1  int = 0, @s_w_id1
smallint = 0, @ol_qty1  smallint = 0,

       @i_id2  int = 0, @s_w_id2
smallint = 0, @ol_qty2  smallint = 0,

       @i_id3  int = 0, @s_w_id3
smallint = 0, @ol_qty3  smallint = 0,

       @i_id4  int = 0, @s_w_id4
smallint = 0, @ol_qty4  smallint = 0,

       @i_id5  int = 0, @s_w_id5
smallint = 0, @ol_qty5  smallint = 0,

       @i_id6  int = 0, @s_w_id6
smallint = 0, @ol_qty6  smallint = 0,

       @i_id7  int = 0, @s_w_id7
smallint = 0, @ol_qty7  smallint = 0,

       @i_id8  int = 0, @s_w_id8
smallint = 0, @ol_qty8  smallint = 0,

       @i_id9  int = 0, @s_w_id9
smallint = 0, @ol_qty9  smallint = 0,

       @i_id10 int = 0, @s_w_id10
smallint = 0, @ol_qty10 smallint = 0,

       @i_id11 int = 0, @s_w_id11
smallint = 0, @ol_qty11 smallint = 0,

       @i_id12 int = 0, @s_w_id12
smallint = 0, @ol_qty12 smallint = 0,

       @i_id13 int = 0, @s_w_id13
smallint = 0, @ol_qty13 smallint = 0,

       @i_id14 int = 0, @s_w_id14
smallint = 0, @ol_qty14 smallint = 0,

       @i_id15 int = 0, @s_w_id15
smallint = 0, @ol_qty15 smallint = 0



as
declare @w_tax          numeric(4,4),
        @d_tax          numeric(4,4),
        @c_last         char(16),
        @c_credit       char(2),
        @c_discount     numeric(4,4),
        @i_price        numeric(5,2),
        @i_name         char(24),
        @i_data         char(50),
        @o_entry_d      datetime,
        @remote_flag    int,
        @s_quantity     smallint,
        @s_data         char(50),
        @s_dist         char(24),
       @li_no          int,
       @o_id          int,
       @commit_flag    tinyint,
        @li_id          int,
        @li_s_w_id      smallint,
        @li_qty         smallint,
       @ol_number     int,
       @c_id_local    int

begin

begin transaction n

--  get district tax and next availible
order id and update
--  plus initialize local variables

       update  district
       set     @d_tax       = d_tax,
               @o_id        =
d_next_o_id,
               d_next_o_id  = d_next_o_id
+ 1,
               @o_entry_d   = getdate(),
               @li_no       = 0,
               @commit_flag = 1
       where   d_w_id       = @w_id and
               d_id         = @d_id


--  process orderlines

       while (@li_no < @o_ol_cnt)
       begin

               select @li_no = @li_no + 1


--  set i_id, s_w_id, and qty for this
lineitem

               select  @li_id = case
@li_no
                                when 1
then @i_id1
                                when 2
then @i_id2
                                when 3
then @i_id3
                                when 4
then @i_id4
                                when 5
then @i_id5
                                when 6
then @i_id6
                                when 7
then @i_id7
                                when 8
then @i_id8
                                when 9
then @i_id9
                                when 10
then @i_id10
```

```
                                when 11
then @i_id11
                                when 12
then @i_id12
                                when 13
then @i_id13
                                when 14
then @i_id14
                                when 15
then @i_id15
                                end,

                       @li_s_w_id = case
@li_no
                                  when 1
then @s_w_id1
                                  when 2
then @s_w_id2
                                  when 3
then @s_w_id3
                                  when 4
then @s_w_id4
                                  when 5
then @s_w_id5
                                  when 6
then @s_w_id6
                                  when 7
then @s_w_id7
                                  when 8
then @s_w_id8
                                  when 9
then @s_w_id9
                                  when
10 then @s_w_id10
                                  when
11 then @s_w_id11
                                  when
12 then @s_w_id12
                                  when
13 then @s_w_id13
                                  when
14 then @s_w_id14
                                  when
15 then @s_w_id15
                                  end,

                       @li_qty = case
@li_no
                                 when 1
then @ol_qty1
                                 when 2
then @ol_qty2
                                 when 3
then @ol_qty3
                                 when 4
then @ol_qty4
                                 when 5
then @ol_qty5
                                 when 6
then @ol_qty6
                                 when 7
then @ol_qty7
                                 when 8
then @ol_qty8
                                 when 9
then @ol_qty9
                                 when 10
then @ol_qty10
                                 when 11
then @ol_qty11
                                 when 12
then @ol_qty12
                                 when 13
then @ol_qty13
                                 when 14
then @ol_qty14
                                 when 15
then @ol_qty15
                                 end

--  get item data (no one updates item)

               select  @i_price =
i_price,
                       @i_name  = i_name,
                       @i_data  = i_data
               from    item (tablock
repeatableread)
               where   i_id = @li_id

--  update stock values

               update  stock
               set     s_ytd          =
s_ytd + @li_qty,
                       @s_quantity    =
s_quantity   = s_quantity - @li_qty +

              case when (s_quantity -
@li_qty < 10) then 91 else 0 end,
                       s_order_cnt    =
s_order_cnt + 1,
                       s_remote_cnt   =
s_remote_cnt + case when (@li_s_w_id =
@w_id) then 0 else 1 end,
                       @s_data        =
s_data,
                       @s_dist        =
case @d_id

when 1  then s_dist_01

when 2  then s_dist_02

when 3  then s_dist_03

when 4  then s_dist_04

when 5  then s_dist_05

when 6  then s_dist_06

when 7  then s_dist_07

when 8  then s_dist_08

when 9  then s_dist_09

when 10 then s_dist_10

end
               where   s_i_id         =
@li_id and
                       s_w_id         =
@li_s_w_id

--  if there actually is a stock (and
item) with these ids, go to work

               if (@@rowcount > 0)
               begin
```

```
--  insert order_line data (using data
from item and stock)

                      insert into
order_line values(@o_id,

             @d_id,

             @w_id,

             @li_no,

             @li_id,

             @li_s_w_id,

             'dec 31, 1899',

             @li_qty,

             @i_price * @li_qty,

             @s_dist)

--  send line-item data to client

                      select  @i_name,

       @s_quantity,
                              b_g = case
when ( (patindex('%ORIGINAL%',@i_data) >
0) and

         (patindex('%ORIGINAL%',@s_data)
> 0) )
                                    then
'B' else 'G' end,
                              @i_price,
                              @i_price *
@li_qty

              end
              else
              begin

--  no item (or stock) found - triggers
rollback condition

                      select '',0,'',0,0
                      select
@commit_flag = 0

              end
       end

--  get customer last name, discount, and
credit rating

       select  @c_last     = c_last,
               @c_discount = c_discount,
               @c_credit   = c_credit,
               @c_id_local = c_id
       from    customer (repeatableread)
       where   c_id          = @c_id and
               c_w_id        = @w_id and
               c_d_id        = @d_id

--  insert fresh row into orders table

       insert into orders values (
       @o_id,

       @d_id,

       @w_id,

       @c_id_local,

       @o_entry_d,
                                 0,

       @o_ol_cnt,

       @o_all_local)

--  insert corresponding row into new-
order table

       insert into new_order values (
       @o_id,

       @d_id,

       @w_id)

--  select warehouse tax

       select  @w_tax  = w_tax
       from    warehouse (repeatableread)
       where   w_id    = @w_id

       if (@commit_flag = 1)

               commit transaction n
       else

--  all that work for nuthin!!!

               rollback transaction n

--  return order data to client

       select  @w_tax,
               @d_tax,
               @o_id,
               @c_last,
               @c_discount,
               @c_credit,
               @o_entry_d,
               @commit_flag

end

go
```

Ordstat.sql

```
--  File:     ORDSTAT.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates order status
transaction stored procedure
--
--      Interface Level: 4.10.000

use tpcc
go
```

```
if exists ( select name from sysobjects
where name = 'tpcc_orderstatus' )
       drop procedure   tpcc_orderstatus
go

create proc tpcc_orderstatus  @w_id
       smallint,
                              @d_id
       tinyint,

                              @c_id
       int,
                              @c_last
       char(16) = ''


as

declare @c_balance
       numeric(12,2),
        @c_first              char(16),
        @c_middle             char(2),
        @o_id                 int,
        @o_entry_d            datetime,
        @o_carrier_id         smallint,
        @cnt                  smallint

begin tran o

if (@c_id = 0)
       begin

--  get customer id and info using last
name

              select  @cnt   =
(count(*)+1)/2
              from    customer
(repeatableread)
              where   c_last  = @c_last
and
                      c_w_id  = @w_id and
                      c_d_id  = @d_id

              set     rowcount @cnt

              select  @c_id          =
c_id,
                      @c_balance     =
c_balance,
                         @c_first    =
c_first,
                         @c_last     =
c_last,
                         @c_middle   =
c_middle
              from    customer
(repeatableread)
              where   c_last         =
@c_last and
                      c_w_id         =
@w_id and
                      c_d_id         =
@d_id
              order   by c_w_id, c_d_id,
c_last, c_first

              set     rowcount 0
       end
       else

       begin

--  get customer info if by id

              select  @c_balance     =
c_balance,
                      @c_first       =
c_first,
                      @c_middle      =
c_middle,
                      @c_last        =
c_last
              from    customer
(repeatableread)
              where   c_id           =
@c_id and
                      c_d_id         =
@d_id and
                      c_w_id         =
@w_id

              select  @cnt   =
@@rowcount

       end

--  if no such customer

       if (@cnt = 0)
       begin
              raiserror('Customer not
found',18,1)
              goto custnotfound
       end

--  get order info

       select  @o_id          = o_id,
               @o_entry_d     =
o_entry_d,
               @o_carrier_id  =
o_carrier_id
       from    orders (serializable)
       where   o_c_id         = @c_id and
               o_d_id         = @d_id and
               o_w_id         = @w_id
       order   by o_id asc

--  select order lines for the current
order

       select  ol_supply_w_id,
               ol_i_id,
               ol_quantity,
               ol_amount,
               ol_delivery_d
       from    order_line
(repeatableread)
       where   ol_o_id = @o_id and
               ol_d_id = @d_id and
               ol_w_id = @w_id

custnotfound:

commit tran o

--  return data to client

select  @c_id,
```

```
        @c_last,
        @c_first,
        @c_middle,

        @o_entry_d,
        @o_carrier_id,
        @c_balance,
        @o_id

go
```

Payment.sql

```
--  File:     PAYMENT.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates payment transaction
stored procedure
--
--      Interface Level: 4.10.000

use tpcc
go

if exists (select name from sysobjects
where name = 'tpcc_payment' )
        drop procedure tpcc_payment
go

create proc tpcc_payment        @w_id
smallint,
                                @c_w_id
smallint,
                                @h_amount
numeric(6,2),
                                @d_id
tinyint,
                                @c_d_id
tinyint,
                                @c_id
int,
                                @c_last
char(16) = ''




as
declare  @w_street_1     char(20),
         @w_street_2     char(20),
         @w_city         char(20),
         @w_state        char(2),
         @w_zip          char(9),
         @w_name         char(10),
         @d_street_1     char(20),
         @d_street_2     char(20),
         @d_city         char(20),
         @d_state        char(2),
         @d_zip          char(9),
         @d_name         char(10),
         @c_first        char(16),
         @c_middle       char(2),
         @c_street_1     char(20),
         @c_street_2     char(20),
         @c_city         char(20),
         @c_state        char(2),
         @c_zip          char(9),
         @c_phone        char(16),
         @c_since        datetime,
         @c_credit       char(2),
         @c_credit_lim   numeric(12,2),
         @c_balance      numeric(12,2),
         @c_discount     numeric(4,4),
         @data           char(500),
         @c_data         char(500),
         @datetime       datetime,
         @w_ytd          numeric(12,2),
         @d_ytd          numeric(12,2),
         @cnt            smallint,
         @val            smallint,
         @screen_data   char(200),
        @d_id_local    tinyint,
        @w_id_local    smallint,
        @c_id_local    int

select @screen_data = ''

begin tran p

--  get payment date

        select  @datetime = getdate()

        if (@c_id = 0)
        begin

--  get customer id and info using last
name

                select  @cnt   = count(*)
                from    customer
(repeatableread)
                where   c_last  = @c_last
and
                        c_w_id  = @c_w_id
and
                        c_d_id  = @c_d_id



                select  @val = (@cnt + 1)
/ 2
                set     rowcount @val

                select  @c_id   = c_id
                from    customer
(repeatableread)
                where   c_last  = @c_last
and
                        c_w_id  = @c_w_id
and
                        c_d_id  = @c_d_id
                order   by c_last, c_first

                set     rowcount 0
        end

--  get customer info and update balances

        update  customer
        set     @c_balance      = c_balance
        = c_balance - @h_amount,
                c_payment_cnt   =
c_payment_cnt + 1,
                c_ytd_payment   =
c_ytd_payment + @h_amount,
                @c_first        = c_first,
                @c_middle       = c_middle,
                @c_last         = c_last,
```

```
                @c_street_1    =
c_street_1,
                @c_street_2    =
c_street_2,
                @c_city        = c_city,
                @c_state       = c_state,
                @c_zip         = c_zip,
                @c_phone       = c_phone,
                @c_credit      = c_credit,
                @c_credit_lim  =
c_credit_lim,
                @c_discount    =
c_discount,
                @c_since       = c_since,

                @data          = c_data,

                @c_id_local    = c_id
        where   c_id           = @c_id and
                c_w_id         = @c_w_id
and
                c_d_id         = @c_d_id

--  if customer has bad credit get some
more info

        if (@c_credit = 'BC')
        begin

--      compute new info

                select @c_data =
convert(char(5),@c_id) +

convert(char(4),@c_d_id) +

convert(char(5),@c_w_id) +

convert(char(4),@d_id) +

convert(char(5),@w_id) +

convert(char(19),@h_amount) +

substring(@data, 1, 458)

--  update customer info

                update  customer
                set     c_data  = @c_data

                where   c_id    = @c_id and
                        c_w_id  = @c_w_id
and
                        c_d_id  = @c_d_id

                select  @screen_data =
substring (@c_data,1,200)
        end

--  get district data and update year-to-
date

        update  district
        set     d_ytd          = d_ytd +
@h_amount,
                @d_street_1    =
d_street_1,
                @d_street_2    =
d_street_2,
                @d_city        = d_city,
                @d_state       = d_state,
                @d_zip         = d_zip,
                @d_name        = d_name,
                @d_id_local    = d_id
        where   d_w_id         = @w_id and
                d_id           = @d_id

--  get warehouse data and update year-
to-date

        update  warehouse
        set     w_ytd          = w_ytd +
@h_amount,
                @w_street_1    =
w_street_1,
                @w_street_2    =
w_street_2,
                @w_city        = w_city,
                @w_state       = w_state,
                @w_zip         = w_zip,
                @w_name        = w_name,
                @w_id_local    = w_id
        where   w_id           = @w_id

--  create history record

        insert into history values (
        @c_id_local,

        @c_d_id,

        @c_w_id,

        @d_id_local,

        @w_id_local,

        @datetime,

        @h_amount,

        @w_name + '    ' + @d_name)
commit tran p

--  return data to client

select  @c_id,
        @c_last,
        @datetime,
        @w_street_1,
        @w_street_2,
        @w_city,

        @w_state,
        @w_zip,
        @d_street_1,
        @d_street_2,
        @d_city,
        @d_state,
        @d_zip,
        @c_first,
        @c_middle,
        @c_street_1,
        @c_street_2,
        @c_city,
        @c_state,
        @c_zip,
        @c_phone,
        @c_since,
        @c_credit,
        @c_credit_lim,
```

```
        @c_discount,
        @c_balance,

        @screen_data

go
```

Delivery.sql

```
--  File:     DELIVERY.SQL
--            Microsoft TPC-C Benchmark
Kit Ver. 4.41
--            Copyright Microsoft, 2001
--  Purpose:  Creates delivery
transaction stored procedure
--
--      Interface Level: 4.10.000

use tpcc
go

if exists (select name from sysobjects
where name = 'tpcc_delivery' )
        drop procedure tpcc_delivery
go

create proc tpcc_delivery     @w_id
smallint,


        @o_carrier_id   smallint
as


declare @d_id  tinyint,
        @o_id  int,
        @c_id  int,
        @total numeric(12,2),
        @oid1  int,
        @oid2  int,
        @oid3  int,
        @oid4  int,
        @oid5  int,
        @oid6  int,
        @oid7  int,
        @oid8  int,
        @oid9  int,
        @oid10 int

select @d_id = 0

begin tran d

        while (@d_id < 10)
        begin

                select @d_id  = @d_id +
1,
                        @total = 0,
                        @o_id  = 0

                select  top 1
                        @o_id   = no_o_id
                from    new_order
(serializable updlock)
                where   no_w_id = @w_id and
                        no_d_id = @d_id
                order   by no_o_id asc

                if (@@rowcount <> 0)
                begin

--  claim the order for this district

                        delete  new_order
                        where   no_w_id =
@w_id and
                                no_d_id =
@d_id and
                                no_o_id =
@o_id

--  set carrier_id on this order (and get
customer id)

                        update  orders
                        set
        o_carrier_id    = @o_carrier_id,
                                @c_id
        = o_c_id
                        where   o_w_id
        = @w_id and
                                o_d_id
        = @d_id and
                                o_id
        = @o_id

--  set date in all lineitems for this
order (and sum amounts)

                        update  order_line
                        set
        ol_delivery_d   = getdate(),
                                @total
        = @total + ol_amount
                        where   ol_w_id
        = @w_id and
                                ol_d_id
        = @d_id and
                                ol_o_id
        = @o_id

--  accummulate lineitem amounts for this
order into customer

                        update  customer
                        set     c_balance
        = c_balance + @total,

        c_delivery_cnt = c_delivery_cnt +
1
                        where   c_w_id
        = @w_id and
                                c_d_id
        = @d_id and
                                c_id
        = @c_id

                end

        select @oid1 = case @d_id when  1
then @o_id else @oid1 end,
                @oid2 = case @d_id when  2
then @o_id else @oid2 end,
                @oid3 = case @d_id when  3
then @o_id else @oid3 end,
                @oid4 = case @d_id when  4
then @o_id else @oid4 end,
                @oid5 = case @d_id when  5
then @o_id else @oid5 end,
```

```
                @oid6 = case @d_id when  6
then @o_id else @oid6 end,
                @oid7 = case @d_id when  7
then @o_id else @oid7 end,
                @oid8 = case @d_id when  8
then @o_id else @oid8 end,
                @oid9 = case @d_id when  9
then @o_id else @oid9 end,
                @oid10 = case @d_id when
10 then @o_id else @oid10 end

    end

commit tran d

-- return delivery data to client

select @oid1,
       @oid2,
       @oid3,
       @oid4,
       @oid5,
       @oid6,
       @oid7,
       @oid8,
       @oid9,
       @oid10

go
```

Loader Source Code

Tpccldr.c

```
//      File:          TPCCLDR.C
//                             Microsoft
TPC-C Kit Ver. 4.41
//                             Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000,
2001
//      Purpose:       Source file for
TPC-C database loader


// Includes
#include "tpcc.h"
#include "search.h"

// Defines
#define MAXITEMS                100000
#define MAXITEMS_SCALE_DOWN          100
#define CUSTOMERS_PER_DISTRICT  3000
#define CUSTOMERS_SCALE_DOWN  30
#define DISTRICT_PER_WAREHOUSE  10
#define ORDERS_PER_DISTRICT     3000
#define ORDERS_SCALE_DOWN       30
#define MAX_CUSTOMER_THREADS    2
#define MAX_ORDER_THREADS       3
#define MAX_MAIN_THREADS        4

// Functions declarations

void HandleErrorDBC (SQLHDBC  hdbc1);

void CheckDataBase();

long NURand();
void LoadItem();
void LoadWarehouse();

void Stock();
void District();

void LoadCustomer();
void CustomerBufInit();
void CustomerBufLoad();
void LoadCustomerTable();
void LoadHistoryTable();

void LoadOrders();
void OrdersBufInit();
void OrdersBufLoad();
void LoadOrdersTable();
void LoadNewOrderTable();
void LoadOrderLineTable();
void GetPermutation();
void CheckForCommit();
void OpenConnections();
void BuildIndex();
void FormatDate ();

// Shared memory structures

typedef struct
{
    long                ol;
    long                ol_i_id;
    short               ol_supply_w_id;
    short               ol_quantity;
    double              ol_amount;
    char
ol_dist_info[DIST_INFO_LEN+1];
        char
        ol_delivery_d[OL_DELIVERY_D_LEN+1
];
} ORDER_LINE_STRUCT;

typedef struct
{
    long                o_id;
    short               o_d_id;
    short               o_w_id;
    long                o_c_id;
    short               o_carrier_id;
    short               o_ol_cnt;
    short               o_all_local;
        ORDER_LINE_STRUCT   o_ol[15];
} ORDERS_STRUCT;

typedef struct
{
    long
        c_id;
    short
        c_d_id;
    short
        c_w_id;
        char
        c_first[FIRST_NAME_LEN+1];
    char
        c_middle[MIDDLE_NAME_LEN+1];
    char
        c_last[LAST_NAME_LEN+1];
    char
        c_street_1[ADDRESS_LEN+1];
    char
        c_street_2[ADDRESS_LEN+1];
```

```
    char
       c_city[ADDRESS_LEN+1];
    char
       c_state[STATE_LEN+1];
    char
       c_zip[ZIP_LEN+1];
    char
       c_phone[PHONE_LEN+1];
    char
       c_credit[CREDIT_LEN+1];
    double
       c_credit_lim;
    double
       c_discount;
//  fix to avoid ODBC float to numeric
conversion problem.
//  double
       c_balance;
       char
       c_balance[6];

    double
       c_ytd_payment;
       short
       c_payment_cnt;
       short
       c_delivery_cnt;
       char
       c_data[C_DATA_LEN+1];
       double
       h_amount;
    char
       h_data[H_DATA_LEN+1];
} CUSTOMER_STRUCT;

typedef struct
{
       char
       c_last[LAST_NAME_LEN+1];
       char
       c_first[FIRST_NAME_LEN+1];
    long
       c_id;
} CUSTOMER_SORT_STRUCT;

typedef struct
{
    long                time_start;
} LOADER_TIME_STRUCT;


// Global variables

char   szLastError[300];

HENV   henv;

HDBC   v_hdbc;
               // for SQL Server version
verification
HDBC   i_hdbc1;
               // for ITEM table
HDBC   w_hdbc1;
               // for WAREHOUSE,
DISTRICT, STOCK
HDBC   c_hdbc1;
               // for CUSTOMER
HDBC   c_hdbc2;
               // for HISTORY
HDBC   o_hdbc1;
               // for ORDERS
HDBC   o_hdbc2;
               // for NEW-ORDER

HDBC   o_hdbc3;
               // for ORDER-LINE

HSTMT  v_hstmt;
               // for SQL Server version
verification
HSTMT  i_hstmt1;
HSTMT  w_hstmt1;
HSTMT  c_hstmt1, c_hstmt2;
HSTMT  o_hstmt1, o_hstmt2, o_hstmt3;

ORDERS_STRUCT
orders_buf[ORDERS_PER_DISTRICT];
CUSTOMER_STRUCT
customer_buf[CUSTOMERS_PER_DISTRICT];
long            orders_rows_loaded;
long            new_order_rows_loaded;
long            order_line_rows_loaded;
long            history_rows_loaded;
long            customer_rows_loaded;
long            stock_rows_loaded;
long            district_rows_loaded;
long            item_rows_loaded;
long            warehouse_rows_loaded;
long            main_time_start;
long            main_time_end;
long                  max_items;
long
       customers_per_district;
long
       orders_per_district;
long                  first_new_order;
long                  last_new_order;

TPCCLDR_ARGS    *aptr, args;

//=======================================
===============================
//
// Function name: main
//
//=======================================
===============================

int main(int  argc, char **argv)
{

       DWORD
dwThreadID[MAX_MAIN_THREADS];
       HANDLE
hThread[MAX_MAIN_THREADS];
       FILE            *fLoader;
       char            buffer[255];
       int                           i;

       for (i=0; i<MAX_MAIN_THREADS;
i++)
              hThread[i] = NULL;


printf("\n*******************************
********************");
       printf("\n*
*");
       printf("\n*  Microsoft SQL Server
*");
       printf("\n*
*");
```

```
	printf("\n*  TPC-C BENCHMARK KIT:
Database loader          *");
	printf("\n*  Version %s
*", TPCKIT_VER);
	printf("\n*
*");
	printf("\n**********************
***************************\n\n");

    // process command line arguments

    aptr = &args;
    GetArgsLoader(argc, argv, aptr);


	// verify database and tables
exist before attempting to load
	//CheckDataBase();


	printf("Build interface is
ODBC.\n");

	if (aptr->build_index == 0)
		printf("Data load only -
no index creation.\n");
	else
		printf("Data load and
index creation.\n");

	if (aptr->index_order == 0)
		printf("Clustered indexes
will be created after bulk load.\n");
	else
		printf("Clustered indexes
will be created before bulk load.\n");

	// set database scale values
	if (aptr->scale_down == 1)
	{
		printf("*** Scaled Down
Database ***\n");
		max_items =
MAXITEMS_SCALE_DOWN;
		customers_per_district =
CUSTOMERS_SCALE_DOWN;
		orders_per_district =
ORDERS_SCALE_DOWN;
		first_new_order = 0;
		last_new_order  = 30;
	}
	else
	{
		max_items = MAXITEMS;
		customers_per_district =
CUSTOMERS_PER_DISTRICT;
		orders_per_district =
ORDERS_PER_DISTRICT;
		first_new_order = 2100;
		last_new_order  = 3000;
	}


	// open connections to SQL Server

	OpenConnections();

	// open file for loader results
	fLoader = fopen(aptr-
>loader_res_file, "w");

    if (fLoader == NULL)
	{
		printf("Error, loader
result file open failed.");
		exit(-1);
	}

    // start loading data

    sprintf(buffer,"TPC-C load started
for %ld warehouses.\n",aptr-
>num_warehouses);

	printf("%s",buffer);
	fprintf(fLoader,"%s",buffer);

	main_time_start = (TimeNow() /
MILLI);

	// start parallel load threads


	if (aptr->tables_all || aptr-
>table_item)
	{
		fprintf(fLoader,
"\nStarting loader threads for: item\n");


		hThread[0] =
CreateThread(NULL,

				0,


(LPTHREAD_START_ROUTINE) LoadItem,

				NULL,

				0,

				&dwThreadID[0]);

		if (hThread[0] == NULL)
		{
			printf("Error,
failed in creating creating thread =
0.\n");
			exit(-1);
		}
	}

	if (aptr->tables_all || aptr-
>table_warehouse)
	{
		fprintf(fLoader, "Starting
loader threads for: warehouse\n");

		hThread[1] =
CreateThread(NULL,

				0,


(LPTHREAD_START_ROUTINE) LoadWarehouse,

				NULL,

				0,
```

```
                        &dwThreadID[1]);

                if (hThread[1] == NULL)
                {
                        printf("Error,
failed in creating creating thread =
1.\n");
                        exit(-1);

                }
        }

        if (aptr->tables_all || aptr-
>table_customer)
        {
                fprintf(fLoader, "Starting
loader threads for: customer\n");

                hThread[2] =
CreateThread(NULL,

                        0,


(LPTHREAD_START_ROUTINE) LoadCustomer,

                        NULL,

                        0,

                        &dwThreadID[2]);

                if (hThread[2] == NULL)
                {
                        printf("Error,
failed in creating creating main thread =
2.\n");
                        exit(-1);
                }
        }

        if (aptr->tables_all || aptr-
>table_orders)
        {
                fprintf(fLoader, "Starting
loader threads for: orders\n");

                hThread[3] =
CreateThread(NULL,

                        0,


(LPTHREAD_START_ROUTINE) LoadOrders,

                        NULL,

                        0,

                        &dwThreadID[3]);

                if (hThread[3] == NULL)
                {
                        printf("Error,
failed in creating creating main thread =
3.\n");
                        exit(-1);
                }
        }

        // Wait for threads to finish...
        for (i=0; i<MAX_MAIN_THREADS;
i++)
        {
                if (hThread[i] != NULL)
                {

        WaitForSingleObject( hThread[i],
INFINITE );

        CloseHandle(hThread[i]);
                        hThread[i] = NULL;
                }
        }

        main_time_end = (TimeNow() /
MILLI);

    sprintf(buffer,"\nTPC-C load
completed successfully in %ld
minutes.\n",
                        (main_time_end -
main_time_start)/60);

        printf("%s",buffer);
        fprintf(fLoader, "%s", buffer);

        fclose(fLoader);

        SQLFreeEnv(henv);

    exit(0);

        return 0;
}


//=======================================
================================
//
// Function name: LoadItem
//
//=======================================
================================

void LoadItem()
{
    long                i_id;
        long            i_im_id;
    char
        i_name[I_NAME_LEN+1];
    double              i_price;
    char
        i_data[I_DATA_LEN+1];
        char            name[20];
        long            time_start;
        RETCODE         rc;
        DBINT           rcint;
        char            bcphint[128];
        char            err_log_path[256];

        // Seed with unique number
        seed(1);

        printf("Loading item
table...\n");

        // if build index before load
        if ((aptr->build_index == 1) &&
(aptr->index_order == 1))
                BuildIndex("idxitmcl");
```

```
	InitString(i_name, I_NAME_LEN+1);
	InitString(i_data, I_DATA_LEN+1);

	sprintf(name, "%s..%s", aptr-
>database, "item");

	//rc = bcp_init(i_hdbc1, name,
NULL, "logs\\item.err", DB_IN);
	strcpy(err_log_path,aptr-
>log_path);
	strcat(err_log_path,"item.err");
	rc = bcp_init(i_hdbc1, name,
NULL, err_log_path , DB_IN);
	if (rc != SUCCEED)
		HandleErrorDBC(i_hdbc1);

	if ((aptr->build_index == 1) &&
(aptr->index_order == 1))
	{
		sprintf(bcphint, "tablock,
order (i_id), ROWS_PER_BATCH = 100000");
		rc = bcp_control(i_hdbc1,
BCPHINTS, (void*) bcphint);
		if (rc != SUCCEED)

	HandleErrorDBC(i_hdbc1);
	}

	rc = bcp_bind(i_hdbc1, (BYTE *)
&i_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
	if (rc != SUCCEED)
		HandleErrorDBC(i_hdbc1);

	rc = bcp_bind(i_hdbc1, (BYTE *)
&i_im_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 2);
	if (rc != SUCCEED)
		HandleErrorDBC(i_hdbc1);

	rc = bcp_bind(i_hdbc1, (BYTE *)
i_name, 0, I_NAME_LEN, NULL, 0, 0, 3);
	if (rc != SUCCEED)
		HandleErrorDBC(i_hdbc1);

	rc = bcp_bind(i_hdbc1, (BYTE *)
&i_price, 0, SQL_VARLEN_DATA, NULL, 0,
SQLFLT8, 4);
	if (rc != SUCCEED)
		HandleErrorDBC(i_hdbc1);

	rc = bcp_bind(i_hdbc1, (BYTE *)
i_data, 0, I_DATA_LEN, NULL, 0, 0, 5);
	if (rc != SUCCEED)
		HandleErrorDBC(i_hdbc1);

	time_start = (TimeNow() / MILLI);

	item_rows_loaded = 0;

	for (i_id = 1; i_id <= max_items;
i_id++)
	{
		i_im_id = RandomNumber(1L,
10000L);

		MakeAlphaString(14, 24,
I_NAME_LEN, i_name);

		i_price = ((float)
RandomNumber(100L, 10000L))/100.0;


	MakeOriginalAlphaString(26, 50,
I_DATA_LEN, i_data, 10);

		rc = bcp_sendrow(i_hdbc1);

		if (rc != SUCCEED)

	HandleErrorDBC(i_hdbc1);

		item_rows_loaded++;
		CheckForCommit(i_hdbc1,
i_hstmt1, item_rows_loaded, "item",
&time_start);
	}

	rcint = bcp_done(i_hdbc1);
	if (rcint < 0)
		HandleErrorDBC(i_hdbc1);


	printf("Finished loading item
table.\n");

	SQLFreeStmt(i_hstmt1, SQL_DROP);
	SQLDisconnect(i_hdbc1);
	SQLFreeConnect(i_hdbc1);

	// if build index after load
	if ((aptr->build_index == 1) &&
(aptr->index_order == 0))
		BuildIndex("idxitmcl");

}


//=====================================
=====================================
//
// Function   : LoadWarehouse
//
// Loads WAREHOUSE table and loads Stock
and District as Warehouses are created
//
//=====================================
=====================================

void LoadWarehouse()
{
    short      w_id;
    char       w_name[W_NAME_LEN+1];
    char       w_street_1[ADDRESS_LEN+1];
    char       w_street_2[ADDRESS_LEN+1];
    char       w_city[ADDRESS_LEN+1];
    char       w_state[STATE_LEN+1];
    char       w_zip[ZIP_LEN+1];
    double     w_tax;
    double     w_ytd;
	char	name[20];
	long	time_start;
	RETCODE	rc;
	DBINT	rcint;
	char	bcphint[128];
	char	err_log_path[256];

	// Seed with unique number
	seed(2);
```

```
	printf("Loading warehouse 
table...\n");

	// if build index before load...
	if ((aptr->build_index == 1) && 
(aptr->index_order == 1))
		BuildIndex("idxwarcl");

	InitString(w_name, W_NAME_LEN+1);
	InitAddress(w_street_1, 
w_street_2, w_city, w_state, w_zip);

	sprintf(name, "%s..%s", aptr-
>database, "warehouse");

	//rc = bcp_init(w_hdbc1, name, 
NULL, "logs\\whouse.err", DB_IN);
	strcpy(err_log_path,aptr-
>log_path);
	strcat(err_log_path,"whouse.err")
;
	rc = bcp_init(w_hdbc1, name, 
NULL, err_log_path, DB_IN);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	if ((aptr->build_index == 1) && 
(aptr->index_order == 1))
	{
		sprintf(bcphint, "tablock, 
order (w_id), ROWS_PER_BATCH = %d", aptr-
>num_warehouses);
		rc = bcp_control(w_hdbc1, 
BCPHINTS, (void*) bcphint);
		if (rc != SUCCEED)

	HandleErrorDBC(w_hdbc1);
	}

	rc = bcp_bind(w_hdbc1, (BYTE *) 
&w_id, 0, SQL_VARLEN_DATA, NULL, 0, 
SQLINT2, 1);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *) 
w_name, 0, W_NAME_LEN, NULL, 0, 0, 2);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *) 
w_street_1, 0, ADDRESS_LEN, NULL, 0, 0, 
3);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *) 
w_street_2, 0, ADDRESS_LEN, NULL, 0, 0, 
4);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *) 
w_city, 0, ADDRESS_LEN, NULL, 0, 0, 5);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *) 
w_state, 0, STATE_LEN, NULL, 0, 0, 6);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *) 
w_zip, 0, ZIP_LEN, NULL, 0, 0, 7);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *) 
&w_tax, 0, SQL_VARLEN_DATA, NULL, 0, 
SQLFLT8, 8);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *) 
&w_ytd, 0, SQL_VARLEN_DATA, NULL, 0, 
SQLFLT8, 9);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	time_start = (TimeNow() / MILLI);

	warehouse_rows_loaded = 0;

	for (w_id = (short)aptr-
>starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
	{
		MakeAlphaString(6,10, 
W_NAME_LEN, w_name);

		MakeAddress(w_street_1, 
w_street_2, w_city, w_state, w_zip);

		w_tax = ((float) 
RandomNumber(0L,2000L))/10000.00;

		w_ytd = 300000.00;

		rc = bcp_sendrow(w_hdbc1);
		if (rc != SUCCEED)

	HandleErrorDBC(w_hdbc1);

		warehouse_rows_loaded++;
		CheckForCommit(w_hdbc1, 
i_hstmt1, warehouse_rows_loaded, 
"warehouse", &time_start);
	}

	rcint = bcp_done(w_hdbc1);
	if (rcint < 0)
		HandleErrorDBC(w_hdbc1);


	printf("Finished loading 
warehouse table.\n");

	// if build index after load...
	if ((aptr->build_index == 1) && 
(aptr->index_order == 0))
		BuildIndex("idxwarcl");

	stock_rows_loaded = 0;
	district_rows_loaded = 0;

	District();
	Stock();

}
```

```
//=======================================
===============================
//
// Function   : District
//
//=======================================
===============================

void District()
{
    short      d_id;
    short      d_w_id;
    char       d_name[D_NAME_LEN+1];
    char       d_street_1[ADDRESS_LEN+1];
    char       d_street_2[ADDRESS_LEN+1];
    char       d_city[ADDRESS_LEN+1];
    char       d_state[STATE_LEN+1];
    char       d_zip[ZIP_LEN+1];
    double     d_tax;
    double     d_ytd;
       char   name[20];
    long       d_next_o_id;
       long   time_start;
       int           w_id;
       RETCODE rc;
       DBINT  rcint;
       char   bcphint[128];
       char   err_log_path[256];

       // Seed with unique number
       seed(4);

       printf("Loading district
table...\n");

       // build index before load
       if ((aptr->build_index == 1) &&
(aptr->index_order == 1))
              BuildIndex("idxdiscl");

       InitString(d_name, D_NAME_LEN+1);
       InitAddress(d_street_1,
d_street_2, d_city, d_state, d_zip);
       sprintf(name, "%s..%s", aptr-
>database, "district");

       //rc = bcp_init(w_hdbc1, name,
NULL, "logs\\district.err", DB_IN);
       strcpy(err_log_path,aptr-
>log_path);
       strcat(err_log_path,"district.err
");
       rc = bcp_init(w_hdbc1, name,
NULL, err_log_path, DB_IN);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       if ((aptr->build_index == 1) &&
(aptr->index_order == 1))
       {
              sprintf(bcphint, "tablock,
order (d_w_id, d_id), ROWS_PER_BATCH =
%u", (aptr->num_warehouses * 10));
              rc = bcp_control(w_hdbc1,
BCPHINTS, (void*) bcphint);
              if (rc != SUCCEED)

       HandleErrorDBC(w_hdbc1);

       }

       rc = bcp_bind(w_hdbc1, (BYTE *)
&d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 1);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
&d_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
d_name, 0, D_NAME_LEN, NULL, 0, 0, 3);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
d_street_1, 0, ADDRESS_LEN, NULL, 0, 0,
4);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
d_street_2, 0, ADDRESS_LEN, NULL, 0, 0,
5);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
d_city, 0, ADDRESS_LEN, NULL, 0, 0, 6);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
d_state, 0, STATE_LEN, NULL, 0, 0, 7);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
d_zip, 0, ZIP_LEN, NULL, 0, 0, 8);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
&d_tax, 0, SQL_VARLEN_DATA, NULL, 0,
SQLFLT8, 9);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
&d_ytd, 0, SQL_VARLEN_DATA, NULL, 0,
SQLFLT8, 10);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       rc = bcp_bind(w_hdbc1, (BYTE *)
&d_next_o_id, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT4, 11);
       if (rc != SUCCEED)
              HandleErrorDBC(w_hdbc1);

       d_ytd  = 30000.0;

       d_next_o_id =
orders_per_district+1;

       time_start = (TimeNow() / MILLI);
```

```
        for (w_id = aptr-
>starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
        {
                d_w_id = w_id;

                for (d_id = 1; d_id <=
DISTRICT_PER_WAREHOUSE; d_id++)
                {

        MakeAlphaString(6,10,D_NAME_LEN,
d_name);


        MakeAddress(d_street_1,
d_street_2, d_city, d_state, d_zip);

                        d_tax = ((float)
RandomNumber(0L,2000L))/10000.00;

                        rc =
bcp_sendrow(w_hdbc1);
                        if (rc != SUCCEED)


        HandleErrorDBC(w_hdbc1);


        district_rows_loaded++;

        CheckForCommit(w_hdbc1, w_hstmt1,
district_rows_loaded, "district",
&time_start);
                }
    }

        rcint = bcp_done(w_hdbc1);
        if (rcint < 0)
                HandleErrorDBC(w_hdbc1);


        printf("Finished loading district
table.\n");

        // if build index after load...
        if ((aptr->build_index == 1) &&
(aptr->index_order == 0))
                BuildIndex("idxdiscl");

    return;
}


//=======================================
===============================
//
// Function   : Stock
//
//=======================================
===============================

void Stock()
{
    long        s_i_id;
    short       s_w_id;
    short       s_quantity;
    char        s_dist_01[S_DIST_LEN+1];
    char        s_dist_02[S_DIST_LEN+1];
    char        s_dist_03[S_DIST_LEN+1];
    char        s_dist_04[S_DIST_LEN+1];
    char        s_dist_05[S_DIST_LEN+1];
    char        s_dist_06[S_DIST_LEN+1];
    char        s_dist_07[S_DIST_LEN+1];
    char        s_dist_08[S_DIST_LEN+1];
    char        s_dist_09[S_DIST_LEN+1];
    char        s_dist_10[S_DIST_LEN+1];
    long        s_ytd;
    short       s_order_cnt;
    short       s_remote_cnt;
    char        s_data[S_DATA_LEN+1];
    short       len;
        char    name[20];
        long    time_start;
        RETCODE rc;
        DBINT   rcint;
        char    bcphint[128];
        char    err_log_path[256];

        // Seed with unique number
        seed(3);

        // if build index before load...
        if ((aptr->build_index == 1) &&
(aptr->index_order == 1))
                BuildIndex("idxstkcl");

        sprintf(name, "%s..%s", aptr-
>database, "stock");

        //rc = bcp_init(w_hdbc1, name,
NULL, "logs\\stock.err", DB_IN);
        strcpy(err_log_path,aptr-
>log_path);
        strcat(err_log_path,"stock.err");
        rc = bcp_init(w_hdbc1, name,
NULL, err_log_path, DB_IN);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        if ((aptr->build_index == 1) &&
(aptr->index_order == 1))
        {
                sprintf(bcphint, "tablock,
order (s_i_id, s_w_id), ROWS_PER_BATCH =
%u", (aptr->num_warehouses * 100000));
                rc = bcp_control(w_hdbc1,
BCPHINTS, (void*) bcphint);
                if (rc != SUCCEED)

        HandleErrorDBC(w_hdbc1);
        }

        rc = bcp_bind(w_hdbc1, (BYTE *)
&s_i_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        bcp_bind(w_hdbc1, (BYTE *)
&s_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *)
&s_quantity, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 3);
        if (rc != SUCCEED)
                HandleErrorDBC(w_hdbc1);

        rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_01, 0, S_DIST_LEN, NULL, 0, 0, 4);
```

```
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_02, 0, S_DIST_LEN, NULL, 0, 0, 5);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_03, 0, S_DIST_LEN, NULL, 0, 0, 6);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_04, 0, S_DIST_LEN, NULL, 0, 0, 7);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_05, 0, S_DIST_LEN, NULL, 0, 0, 8);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_06, 0, S_DIST_LEN, NULL, 0, 0, 9);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_07, 0, S_DIST_LEN, NULL, 0, 0,
10);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_08, 0, S_DIST_LEN, NULL, 0, 0,
11);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_09, 0, S_DIST_LEN, NULL, 0, 0,
12);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_dist_10, 0, S_DIST_LEN, NULL, 0, 0,
13);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
&s_ytd, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 14);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
&s_order_cnt, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 15);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
&s_remote_cnt, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 16);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = bcp_bind(w_hdbc1, (BYTE *)
s_data, 0, S_DATA_LEN, NULL, 0, 0, 17);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	s_ytd = s_order_cnt =
s_remote_cnt = 0;

	time_start = (TimeNow() / MILLI);

	printf("...Loading stock
table\n");

	for (s_i_id=1; s_i_id <=
max_items; s_i_id++)
	{

		for (s_w_id = (short)aptr-
>starting_warehouse; s_w_id <= aptr-
>num_warehouses; s_w_id++)
		{

			s_quantity =
(short)RandomNumber(10L,100L);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_01);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_02);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_03);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_04);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_05);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_06);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_07);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_08);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_09);
			len =
MakeAlphaString(24,24,S_DIST_LEN,
s_dist_10);


			len =
MakeOriginalAlphaString(26,50,
S_DATA_LEN, s_data,10);


			rc =
bcp_sendrow(w_hdbc1);
			if (rc != SUCCEED)


	HandleErrorDBC(w_hdbc1);


	stock_rows_loaded++;

	CheckForCommit(w_hdbc1, w_hstmt1,
stock_rows_loaded, "stock", &time_start);
```

```
		}
	}

	rcint = bcp_done(w_hdbc1);
	if (rcint < 0)
		HandleErrorDBC(w_hdbc1);


	printf("Finished loading stock 
table.\n");

	SQLFreeStmt(w_hstmt1, SQL_DROP);
	SQLDisconnect(w_hdbc1);
	SQLFreeConnect(w_hdbc1);

	// if build index after load...
	if ((aptr->build_index == 1) && 
(aptr->index_order == 0))
		BuildIndex("idxstkcl");

	return;
}


//=======================================
================================
//
// Function   : LoadCustomer
//
//=======================================
================================

void LoadCustomer()
{
	LOADER_TIME_STRUCT
customer_time_start;
	LOADER_TIME_STRUCT
history_time_start;
	short
	w_id;
    short                     d_id;
	DWORD
	dwThreadID[MAX_CUSTOMER_THREADS];
	HANDLE
	hThread[MAX_CUSTOMER_THREADS];
	char
	name[20];
	RETCODE
	rc;
	DBINT
	rcint;
	char
	bcphint[128];
	char
	cmd[256];
	int
		num_procs;
	char
	err_log_path_cust[256];
	char
	err_log_path_hist[256];
	// SQLRETURN
	rc_1;
	// SQLSMALLINT
	recnum, MsgLen;
	// SQLCHAR
		SqlState[6], 
Msg[SQL_MAX_MESSAGE_LENGTH];
	// SQLINTEGER
	NativeError;

	// Seed with unique number
	seed(5);

	printf("Loading customer and 
history tables...\n");

	// if build index before load...
	if ((aptr->build_index == 1) && 
(aptr->index_order == 1))
	{
		BuildIndex("idxcuscl");
		// check the number of 
processors on this system
		// if 8 or more 
processors, then build index on History.
		// if less than 8 
processors, do not build the index
		num_procs = atoi(getenv( 
"NUMBER_OF_PROCESSORS" ));
		if ( num_procs >= 8 )

	BuildIndex("idxhiscl");
	}

	// Initialize bulk copy
	sprintf(name, "%s..%s", aptr-
>database, "customer");

	//rc = bcp_init(c_hdbc1, name, 
NULL, "logs\\customer.err", DB_IN);
	strcpy(err_log_path_cust,aptr-
>log_path);
	strcat(err_log_path_cust,"custome
r.err");
	rc = bcp_init(c_hdbc1, name, 
NULL, err_log_path_cust, DB_IN);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

	if ((aptr->build_index == 1) && 
(aptr->index_order == 1))
	{
		sprintf(bcphint, "tablock, 
order (c_w_id, c_d_id, c_id), 
ROWS_PER_BATCH = %u", (aptr-
>num_warehouses * 30000));
		rc = bcp_control(c_hdbc1, 
BCPHINTS, (void*) bcphint);
		if (rc != SUCCEED)

	HandleErrorDBC(c_hdbc1);
	}

	sprintf(name, "%s..%s", aptr-
>database, "history");

	rc = bcp_init(c_hdbc2, name, 
NULL, "logs\\history.err", DB_IN);
	strcpy(err_log_path_hist,aptr-
>log_path);
	strcat(err_log_path_hist,"history
.err");
	rc = bcp_init(c_hdbc2, name, 
NULL, err_log_path_hist, DB_IN);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc2);

	sprintf(bcphint, "tablock");
	rc = bcp_control(c_hdbc2, 
BCPHINTS, (void*) bcphint);
```

```
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc2);

	customer_rows_loaded	= 0;
	history_rows_loaded	= 0;

	CustomerBufInit();

	customer_time_start.time_start = 
(TimeNow() / MILLI);
	history_time_start.time_start = 
(TimeNow() / MILLI);

	for (w_id = (short)aptr-
>starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
	{
		for (d_id = 1; d_id <= 
DISTRICT_PER_WAREHOUSE; d_id++)
		{

	CustomerBufLoad(d_id, w_id);

			// Start parallel 
loading threads here...

			// Start customer 
table thread

			printf("...Loading 
customer table for: d_id = %d, w_id = 
%d\n", d_id, w_id);

			hThread[0] = 
CreateThread(NULL,

					0,


(LPTHREAD_START_ROUTINE) 
LoadCustomerTable,


&customer_time_start,

					0,


&dwThreadID[0]);

			if (hThread[0] == 
NULL)
			{

	printf("Error, failed in creating 
creating thread = 0.\n");
				exit(-1);
			}

			// Start History 
table thread

			printf("...Loading 
history table for: d_id = %d, w_id = 
%d\n", d_id, w_id);

			hThread[1] = 
CreateThread(NULL,

					0,

(LPTHREAD_START_ROUTINE) 
LoadHistoryTable,


&history_time_start,

					0,


&dwThreadID[1]);

			if (hThread[1] == 
NULL)
			{

	printf("Error, failed in creating 
creating thread = 1.\n");
				exit(-1);
			}


	WaitForSingleObject( hThread[0], 
INFINITE );

	WaitForSingleObject( hThread[1], 
INFINITE );

			if 
(CloseHandle(hThread[0]) == FALSE)
			{

	printf("Error, failed in closing 
customer thread handle with errno: %d\n", 
GetLastError());
			}

			if 
(CloseHandle(hThread[1]) == FALSE)
			{

	printf("Error, failed in closing 
history thread handle with errno: %d\n", 
GetLastError());
			}


		}

	}

	// flush the bulk connection
	rcint = bcp_done(c_hdbc1);
	if (rcint < 0)
		HandleErrorDBC(c_hdbc1);


	rcint = bcp_done(c_hdbc2);
	if (rcint < 0)
		HandleErrorDBC(c_hdbc2);


	printf("Finished loading customer 
table.\n");

	// if build index after load...
	if ((aptr->build_index == 1) && 
(aptr->index_order == 0))
	{
		BuildIndex("idxcuscl");
```

```
		// check the number of
processors on this system
		// if 8 or more
processors, then build index on History.
		// if less than 8
processors, do not build the index
		num_procs = atoi(getenv(
"NUMBER_OF_PROCESSORS" ));
		if (num_procs >= 8)

	BuildIndex("idxhiscl");
	}

	// build non-clustered index
	if (aptr->build_index == 1)
		BuildIndex("idxcusnc");

	// Output the NURAND used for the
loader into C_FIRST for C_ID = 1,
	// C_W_ID = 1, and C_D_ID = 1
	//sprintf(cmd, "osql -S%s -U%s -
P%s -d%s -e -Q\"update customer set
c_first = 'C_LOAD = %d' where c_id = 1
and c_w_id = 1 and c_d_id = 1\" >
logs\\nurand_load.log",
	sprintf(cmd, "osql -S%s -U%s -P%s
-d%s -e -Q\"update customer set c_first =
'C_LOAD = %d' where c_id = 1 and c_w_id =
1 and c_d_id = 1\" > %snurand_load.log",
				aptr-
>server,
				aptr->user,
				aptr-
>password,
				aptr-
>database,

	LOADER_NURAND_C,
				aptr-
>log_path);

	system(cmd);

	SQLFreeStmt(c_hstmt1, SQL_DROP);
	SQLDisconnect(c_hdbc1);
	SQLFreeConnect(c_hdbc1);

	SQLFreeStmt(c_hstmt2, SQL_DROP);
	SQLDisconnect(c_hdbc2);
	SQLFreeConnect(c_hdbc2);


    return;

}


//=======================================
===============================
//
// Function   : CustomerBufInit
//
//=======================================
===============================


void CustomerBufInit()
{
	int	i;

	for
(i=0;i<customers_per_district;i++)
	{
		customer_buf[i].c_id = 0;
		customer_buf[i].c_d_id =
0;
		customer_buf[i].c_w_id =
0;


	strcpy(customer_buf[i].c_first,""
);

	strcpy(customer_buf[i].c_middle,"
");

	strcpy(customer_buf[i].c_last,"")
;

	strcpy(customer_buf[i].c_street_1
,"");

	strcpy(customer_buf[i].c_street_2
,"");

	strcpy(customer_buf[i].c_city,"")
;

	strcpy(customer_buf[i].c_state,""
);

	strcpy(customer_buf[i].c_zip,"");

	strcpy(customer_buf[i].c_phone,""
);

	strcpy(customer_buf[i].c_credit,"
");


	customer_buf[i].c_credit_lim = 0;
		customer_buf[i].c_discount
= (float) 0;

		//  fix to avoid ODBC
float to numeric conversion problem.
		//
	customer_buf[i].c_balance = 0;

	strcpy(customer_buf[i].c_balance,
"");


	customer_buf[i].c_ytd_payment =
0;

	customer_buf[i].c_payment_cnt =
0;

	customer_buf[i].c_delivery_cnt =
0;


	strcpy(customer_buf[i].c_data,"")
;

		customer_buf[i].h_amount =
0;
```

```
	strcpy(customer_buf[i].h_data,"")
;

	}


}


//=======================================
===============================
//
// Function   : CustomerBufLoad
//
// Fills shared buffer for HISTORY and
CUSTOMER
//=======================================
===============================

void CustomerBufLoad(int d_id, int w_id)
{
	long
	i;
    CUSTOMER_SORT_STRUCT
c[CUSTOMERS_PER_DISTRICT];


	for
(i=0;i<customers_per_district;i++)
	{
		if (i < 1000)
			LastName(i,
c[i].c_last);
		else

	LastName(NURand(255,0,999,LOADER_
NURAND_C), c[i].c_last);


	MakeAlphaString(8,16,FIRST_NAME_L
EN, c[i].c_first);

		c[i].c_id = i+1;

	}

	printf("...Loading customer
buffer for: d_id = %d, w_id = %d\n",
			d_id, w_id);

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

		customer_buf[i].c_d_id =
d_id;
		customer_buf[i].c_w_id =
w_id;
		customer_buf[i].h_amount =
10.0;


	customer_buf[i].c_ytd_payment  =
10.0;


	customer_buf[i].c_payment_cnt  =
1;

	customer_buf[i].c_delivery_cnt =
0;

		// Generate CUSTOMER and
HISTORY data

		customer_buf[i].c_id =
c[i].c_id;


	strcpy(customer_buf[i].c_first,
c[i].c_first);

	strcpy(customer_buf[i].c_last,
c[i].c_last);



	customer_buf[i].c_middle[0] =
'O';

	customer_buf[i].c_middle[1] =
'E';


	MakeAddress(customer_buf[i].c_str
eet_1,

customer_buf[i].c_street_2,

customer_buf[i].c_city,

customer_buf[i].c_state,

customer_buf[i].c_zip);

		MakeNumberString(16, 16,
PHONE_LEN, customer_buf[i].c_phone);

		if (RandomNumber(1L, 100L)
> 10)

	customer_buf[i].c_credit[0] =
'G';
		else

	customer_buf[i].c_credit[0] =
'B';

	customer_buf[i].c_credit[1] =
'C';


	customer_buf[i].c_credit_lim =
50000.0;
		customer_buf[i].c_discount
= ((float) RandomNumber(0L, 5000L)) /
10000.0;

		//  fix to avoid ODBC
float to numeric conversion problem.

		//
customer_buf[i].c_balance = -10.0;

	strcpy(customer_buf[i].c_balance,
"-10.0");
```

```
		MakeAlphaString(300, 500,
C_DATA_LEN, customer_buf[i].c_data);


		// Generate HISTORY data
		MakeAlphaString(12, 24,
H_DATA_LEN, customer_buf[i].h_data);

	}
}

//=======================================
===============================
//
// Function   : LoadCustomerTable
//
//=======================================
===============================

void LoadCustomerTable(LOADER_TIME_STRUCT
*customer_time_start)
{
	int		i;
    long	c_id;
    short	c_d_id;
    short	c_w_id;
    char
c_first[FIRST_NAME_LEN+1];
    char
c_middle[MIDDLE_NAME_LEN+1];
    char	c_last[LAST_NAME_LEN+1];
    char
c_street_1[ADDRESS_LEN+1];
    char
c_street_2[ADDRESS_LEN+1];
    char	c_city[ADDRESS_LEN+1];
    char	c_state[STATE_LEN+1];
    char	c_zip[ZIP_LEN+1];
    char	c_phone[PHONE_LEN+1];
    char	c_credit[CREDIT_LEN+1];
    double	c_credit_lim;
    double	c_discount;

	//  fix to avoid ODBC float to
numeric conversion problem.
	//  double		c_balance;
	char		c_balance[6];

    double	c_ytd_payment;
    short	c_payment_cnt;
    short	c_delivery_cnt;
    char	c_data[C_DATA_LEN+1];
	char
	c_since[C_SINCE_LEN+1];
	RETCODE		rc;

    rc = bcp_bind(c_hdbc1, (BYTE *)
&c_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
&c_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

	rc = bcp_bind(c_hdbc1, (BYTE *)
&c_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 3);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_first, 0, FIRST_NAME_LEN, NULL, 0, 0,
4);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_middle, 0, MIDDLE_NAME_LEN,NULL, 0, 0,
5);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_last, 0, LAST_NAME_LEN, NULL, 0, 0, 6);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_street_1, 0, ADDRESS_LEN, NULL, 0, 0,
7);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_street_2, 0, ADDRESS_LEN, NULL,0,0, 8);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_city, 0, ADDRESS_LEN, NULL, 0, 0, 9);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_state, 0, STATE_LEN, NULL, 0, 0, 10);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_zip, 0, ZIP_LEN, NULL, 0, 0, 11);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_phone, 0, PHONE_LEN, NULL, 0, 0, 12);

	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

	rc = bcp_bind(c_hdbc1, (BYTE *)
&c_since, 0, C_SINCE_LEN, NULL, 0,
SQLCHARACTER, 13);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_credit, 0, CREDIT_LEN, NULL, 0, 0, 14);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
&c_credit_lim, 0, SQL_VARLEN_DATA, NULL,
0, SQLFLT8, 15);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);
```

```
    rc = bcp_bind(c_hdbc1, (BYTE *)
&c_discount, 0, SQL_VARLEN_DATA, NULL, 0,
SQLFLT8, 16);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc1);


       //  fix to avoid ODBC float to
numeric conversion problem.

    //  rc = bcp_bind(c_hdbc1, (BYTE *)
&c_balance, 0, SQL_VARLEN_DATA, NULL, 0,
SQLFLT8, 17);
       //  if (rc != SUCCEED)
       //     HandleErrorDBC(c_hdbc1);
    rc = bcp_bind(c_hdbc1, (BYTE *)
c_balance, 0, 5, NULL, 0, SQLCHARACTER,
17);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc1);


    rc = bcp_bind(c_hdbc1, (BYTE *)
&c_ytd_payment, 0, SQL_VARLEN_DATA, NULL,
0, SQLFLT8, 18);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
&c_payment_cnt, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 19);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
&c_delivery_cnt,0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 20);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc1);

    rc = bcp_bind(c_hdbc1, (BYTE *)
c_data, 0, 500, NULL, 0, 0, 21);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc1);

    for (i = 0; i <
customers_per_district; i++)
    {
              c_id =
customer_buf[i].c_id;
              c_d_id =
customer_buf[i].c_d_id;
              c_w_id =
customer_buf[i].c_w_id;

              strcpy(c_first,
customer_buf[i].c_first);
              strcpy(c_middle,
customer_buf[i].c_middle);
              strcpy(c_last,
customer_buf[i].c_last);
              strcpy(c_street_1,
customer_buf[i].c_street_1);
              strcpy(c_street_2,
customer_buf[i].c_street_2);
              strcpy(c_city,
customer_buf[i].c_city);
              strcpy(c_state,
customer_buf[i].c_state);
              strcpy(c_zip,
customer_buf[i].c_zip);
              strcpy(c_phone,
customer_buf[i].c_phone);
              strcpy(c_credit,
customer_buf[i].c_credit);

              FormatDate(&c_since);


              c_credit_lim =
customer_buf[i].c_credit_lim;
              c_discount =
customer_buf[i].c_discount;

              //  fix to avoid ODBC
float to numeric conversion problem.

              //  c_balance =
customer_buf[i].c_balance;
              strcpy(c_balance,
customer_buf[i].c_balance);

              c_ytd_payment =
customer_buf[i].c_ytd_payment;
              c_payment_cnt =
customer_buf[i].c_payment_cnt;
              c_delivery_cnt =
customer_buf[i].c_delivery_cnt;

              strcpy(c_data,
customer_buf[i].c_data);

              // Send data to server
              rc = bcp_sendrow(c_hdbc1);
              if (rc != SUCCEED)

       HandleErrorDBC(c_hdbc1);


              customer_rows_loaded++;
              CheckForCommit(c_hdbc1,
c_hstmt1, customer_rows_loaded,
"customer", &customer_time_start-
>time_start);
    }

}


//=======================================
================================
//
// Function   : LoadHistoryTable
//
//=======================================
================================

void LoadHistoryTable(LOADER_TIME_STRUCT
*history_time_start)
{
       int         i;
    long        c_id;
    short       c_d_id;
    short       c_w_id;
       double        h_amount;
    char        h_data[H_DATA_LEN+1];
       char
       h_date[H_DATE_LEN+1];
       RETCODE       rc;
```

```
    rc = bcp_bind(c_hdbc2, (BYTE *)
&c_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *)
&c_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *)
&c_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 3);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *)
&c_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 4);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *)
&c_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 5);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc2);

       rc = bcp_bind(c_hdbc2, (BYTE *)
&h_date, 0, H_DATE_LEN, NULL, 0,
SQLCHARACTER, 6);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *)
&h_amount, 0, SQL_VARLEN_DATA, NULL, 0,
SQLFLT8, 7);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc2);

    rc = bcp_bind(c_hdbc2, (BYTE *)
h_data, 0, H_DATA_LEN, NULL, 0, 0, 8);
       if (rc != SUCCEED)
              HandleErrorDBC(c_hdbc2);

    for (i = 0; i <
customers_per_district; i++)
    {
              c_id =
customer_buf[i].c_id;
              c_d_id =
customer_buf[i].c_d_id;
              c_w_id =
customer_buf[i].c_w_id;
              h_amount =
customer_buf[i].h_amount;
              strcpy(h_data,
customer_buf[i].h_data);


              FormatDate(&h_date);


              // send to server
              rc = bcp_sendrow(c_hdbc2);
              if (rc != SUCCEED)
       HandleErrorDBC(c_hdbc2);


              history_rows_loaded++;
              CheckForCommit(c_hdbc2,
c_hstmt2, history_rows_loaded, "history",
&history_time_start->time_start);
    }

}


//=======================================
=====================-===============
//
// Function   : LoadOrders
//
//=======================================
====================================

void LoadOrders()
{
       LOADER_TIME_STRUCT
orders_time_start;
       LOADER_TIME_STRUCT
new_order_time_start;
       LOADER_TIME_STRUCT
order_line_time_start;
       short
       w_id;
    short                    d_id;
       DWORD
       dwThreadID[MAX_ORDER_THREADS];
       HANDLE
       hThread[MAX_ORDER_THREADS];
       char
       name[20];
       RETCODE
       rc;
       char
       bcphint[128];
       char
       err_log_path_ord[256];
       char
       err_log_path_nord[256];
       char
       err_log_path_ordl[256];

       // seed with unique number
       seed(6);

       printf("Loading orders...\n");

       // if build index before load...
       if ((aptr->build_index == 1) &&
(aptr->index_order == 1))
       {

       BuildIndex("idxordcl");

       BuildIndex("idxnodcl");

       BuildIndex("idxodlcl");
       }

       // initialize bulk copy
       sprintf(name, "%s..%s", aptr-
>database, "orders");
```

```
	rc = bcp_init(o_hdbc1, name, 
NULL, "logs\\orders.err", DB_IN);
	strcpy(err_log_path_ord,aptr-
>log_path);
	strcat(err_log_path_ord,"orders.e
rr");
	rc = bcp_init(o_hdbc1, name, 
NULL, err_log_path_ord, DB_IN);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc1);

	if ((aptr->build_index == 1) && 
(aptr->index_order == 1))
	{
		sprintf(bcphint, "tablock, 
order (o_w_id, o_d_id, o_id), 
ROWS_PER_BATCH = %u", (aptr-
>num_warehouses * 30000));
		rc = bcp_control(o_hdbc1, 
BCPHINTS, (void*) bcphint);
		if (rc != SUCCEED)

	HandleErrorDBC(o_hdbc1);
	}

	sprintf(name, "%s..%s", aptr-
>database, "new_order");

	rc = bcp_init(o_hdbc2, name, 
NULL, "logs\\neword.err", DB_IN);
	strcpy(err_log_path_nord,aptr-
>log_path);
	strcat(err_log_path_nord,"neword.
err");
	rc = bcp_init(o_hdbc2, name, 
NULL, err_log_path_nord, DB_IN);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc2);

	if ((aptr->build_index == 1) && 
(aptr->index_order == 1))
	{
		sprintf(bcphint, "tablock, 
order (no_w_id, no_d_id, no_o_id), 
ROWS_PER_BATCH = %u", (aptr-
>num_warehouses * 9000));
		rc = bcp_control(o_hdbc2, 
BCPHINTS, (void*) bcphint);
		if (rc != SUCCEED)

	HandleErrorDBC(o_hdbc2);
	}

	sprintf(name, "%s..%s", aptr-
>database, "order_line");

	rc = bcp_init(o_hdbc3, name, 
NULL, "logs\\ordline.err", DB_IN);
	strcpy(err_log_path_ordl,aptr-
>log_path);
	strcat(err_log_path_ordl,"ordline
.err");
	rc = bcp_init(o_hdbc3, name, 
NULL, err_log_path_ordl, DB_IN);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    if ((aptr->build_index == 1) && 
(aptr->index_order == 1))
	{
		sprintf(bcphint, "tablock, 
order (ol_w_id, ol_d_id, ol_o_id, 
ol_number), ROWS_PER_BATCH = %u", (aptr-
>num_warehouses * 300000));
		rc = bcp_control(o_hdbc3, 
BCPHINTS, (void*) bcphint);
		if (rc != SUCCEED)

	HandleErrorDBC(o_hdbc3);
	}

	orders_rows_loaded      = 0;
	new_order_rows_loaded   = 0;
	order_line_rows_loaded  = 0;

	OrdersBufInit();

	orders_time_start.time_start = 
(TimeNow() / MILLI);
	new_order_time_start.time_start = 
(TimeNow() / MILLI);
	order_line_time_start.time_start 
= (TimeNow() / MILLI);

	for (w_id = (short)aptr-
>starting_warehouse; w_id <= aptr-
>num_warehouses; w_id++)
	{
		for (d_id = 1; d_id <= 
DISTRICT_PER_WAREHOUSE; d_id++)
		{


	OrdersBufLoad(d_id, w_id);

			// start parallel 
loading threads here...

			// start Orders 
table thread

			printf("...Loading 
Order Table for: d_id = %d, w_id = %d\n", 
d_id, w_id);

			hThread[0] = 
CreateThread(NULL,
						0,

(LPTHREAD_START_ROUTINE) LoadOrdersTable,

&orders_time_start,
						0,

&dwThreadID[0]);

			if (hThread[0] == 
NULL)
			{

	printf("Error, failed in creating 
creating thread = 0.\n");
				exit(-1);
			}
```

```
                        // start NewOrder
table thread

                        printf("...Loading
New-Order Table for: d_id = %d, w_id =
%d\n", d_id, w_id);

                        hThread[1] =
CreateThread(NULL,

                                  0,

(LPTHREAD_START_ROUTINE)
LoadNewOrderTable,

&new_order_time_start,

                                  0,

&dwThreadID[1]);

                        if (hThread[1] ==
NULL)
                        {

        printf("Error, failed in creating
creating thread = 1.\n");
                                exit(-1);
                        }

                        // start Order-
Line table thread

                        printf("...Loading
Order-Line Table for: d_id = %d, w_id =
%d\n", d_id, w_id);

                        hThread[2] =
CreateThread(NULL,

                                  0,

(LPTHREAD_START_ROUTINE)
LoadOrderLineTable,

&order_line_time_start,

                                  0,

&dwThreadID[2]);

                        if (hThread[2] ==
NULL)
                        {

        printf("Error, failed in creating
creating thread = 2.\n");
                                exit(-1);
                        }


        WaitForSingleObject( hThread[0],
INFINITE );

        WaitForSingleObject( hThread[1],
INFINITE );

        WaitForSingleObject( hThread[2],
INFINITE );

                        if
(CloseHandle(hThread[0]) == FALSE)
                        {

        printf("Error, failed in closing
Orders thread handle with errno: %d\n",
GetLastError());
                        }

                        if
(CloseHandle(hThread[1]) == FALSE)
                        {

        printf("Error, failed in closing
NewOrder thread handle with errno: %d\n",
GetLastError());
                        }

                        if
(CloseHandle(hThread[2]) == FALSE)
                        {

        printf("Error, failed in closing
OrderLine thread handle with errno:
%d\n", GetLastError());
                        }

                }
        }

        printf("Finished loading
orders.\n");


    return;
}


//=======================================
================================
//
// Function   : OrdersBufInit
//
// Clears shared buffer for ORDERS,
NEWORDER, and ORDERLINE
//
//=======================================
================================

void OrdersBufInit()
{
        int     i;
        int             j;

        for
(i=0;i<orders_per_district;i++)
        {
                orders_buf[i].o_id = 0;
                orders_buf[i].o_d_id = 0;
                orders_buf[i].o_w_id = 0;
                orders_buf[i].o_c_id = 0;
                orders_buf[i].o_carrier_id
= 0;
```

```
		orders_buf[i].o_ol_cnt =
0;
		orders_buf[i].o_all_local
= 0;

		for (j=0;j<=14;j++)
		{

	orders_buf[i].o_ol[j].ol = 0;

	orders_buf[i].o_ol[j].ol_i_id =
0;

	orders_buf[i].o_ol[j].ol_supply_w
_id = 0;

	orders_buf[i].o_ol[j].ol_quantity
= 0;

	orders_buf[i].o_ol[j].ol_amount =
0;

	strcpy(orders_buf[i].o_ol[j].ol_d
ist_info,"");
		}
	}



}


//=======================================
================================
//
// Function   : OrdersBufLoad
//
// Fills shared buffer for ORDERS,
NEWORDER, and ORDERLINE
//
//=======================================
================================

void OrdersBufLoad(int d_id, int w_id)
{

	int
cust[ORDERS_PER_DISTRICT+1];
	long    o_id;
	short   ol;

	printf("...Loading Order Buffer
for: d_id = %d, w_id = %d\n",
			d_id, w_id);

	GetPermutation(cust,
orders_per_district);

	for
(o_id=0;o_id<orders_per_district;o_id++)
	{

		// Generate ORDER and NEW-
ORDER data

		orders_buf[o_id].o_d_id =
d_id;
		orders_buf[o_id].o_w_id =
w_id;
		orders_buf[o_id].o_id =
o_id+1;
		orders_buf[o_id].o_c_id =
cust[o_id+1];
		orders_buf[o_id].o_ol_cnt
= (short)RandomNumber(5L, 15L);

		if (o_id <
first_new_order)
		{

	orders_buf[o_id].o_carrier_id =
(short)RandomNumber(1L, 10L);

	orders_buf[o_id].o_all_local  =
1;
		}
		else
		{

	orders_buf[o_id].o_carrier_id =
0;

	orders_buf[o_id].o_all_local  =
1;
		}

		for (ol=0;
ol<orders_buf[o_id].o_ol_cnt; ol++)
		{


	orders_buf[o_id].o_ol[ol].ol =
ol+1;

	orders_buf[o_id].o_ol[ol].ol_i_id
= RandomNumber(1L, max_items);

	orders_buf[o_id].o_ol[ol].ol_supp
ly_w_id = w_id;

	orders_buf[o_id].o_ol[ol].ol_quan
tity = 5;

	MakeAlphaString(24, 24,
OL_DIST_INFO_LEN,
&orders_buf[o_id].o_ol[ol].ol_dist_info);

			// Generate ORDER-
LINE data
			if (o_id <
first_new_order)
			{

	orders_buf[o_id].o_ol[ol].ol_amou
nt = 0;
				// Added to
insure ol_delivery_d set properly during
load


	FormatDate(&orders_buf[o_id].o_ol
[ol].ol_delivery_d);


			}
			else
			{

	orders_buf[o_id].o_ol[ol].ol_amou
nt = RandomNumber(1,999999)/100.0;
```

```
                              // Added to
insure ol_delivery_d set properly during
load

                              // odbc
datetime format

        strcpy(orders_buf[o_id].o_ol[ol].
ol_delivery_d,"1899-12-31 00:00:00.000");


                        }
                }
        }
}


//=======================================
===============================
//
// Function   : LoadOrdersTable
//
//=======================================
===============================

void LoadOrdersTable(LOADER_TIME_STRUCT
*orders_time_start)
{
        int         i;
    long        o_id;
        short          o_d_id;
        short          o_w_id;
    long        o_c_id;
    short       o_carrier_id;
    short       o_ol_cnt;
    short       o_all_local;
        char
        o_entry_d[O_ENTRY_D_LEN+1];
        RETCODE        rc;
        DBINT          rcint;

        // bind ORDER data
    rc = bcp_bind(o_hdbc1, (BYTE *)
&o_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *)
&o_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *)
&o_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 3);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *)
&o_c_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 4);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

        rc = bcp_bind(o_hdbc1, (BYTE *)
&o_entry_d, 0, O_ENTRY_D_LEN, NULL, 0,
SQLCHARACTER, 5);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *)
&o_carrier_id, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 6);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *)
&o_ol_cnt, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 7);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

    rc = bcp_bind(o_hdbc1, (BYTE *)
&o_all_local, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 8);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

    for (i = 0; i < orders_per_district;
i++)
    {
                o_id         =
orders_buf[i].o_id;
                o_d_id       =
orders_buf[i].o_d_id;
                o_w_id       =
orders_buf[i].o_w_id;
                o_c_id       =
orders_buf[i].o_c_id;
                o_carrier_id =
orders_buf[i].o_carrier_id;
                o_ol_cnt     =
orders_buf[i].o_ol_cnt;
                o_all_local  =
orders_buf[i].o_all_local;

                FormatDate(&o_entry_d);

                // send data to server
                rc = bcp_sendrow(o_hdbc1);
                if (rc != SUCCEED)


        HandleErrorDBC(o_hdbc1);


                orders_rows_loaded++;

                CheckForCommit(o_hdbc1,
o_hstmt1, orders_rows_loaded, "orders",
&orders_time_start->time_start);
    }

        // rcint = bcp_batch(o_hdbc1);
        // if (rcint < 0)
        //      HandleErrorDBC(o_hdbc1);


        if ((o_w_id == aptr-
>num_warehouses) && (o_d_id == 10))
        {
                rcint = bcp_done(o_hdbc1);

                if (rcint < 0)

        HandleErrorDBC(o_hdbc1);


                SQLFreeStmt(o_hstmt1,
SQL_DROP);
```

```
		SQLDisconnect(o_hdbc1);
		SQLFreeConnect(o_hdbc1);

		// if build index after
load...
		if ((aptr->build_index ==
1) && (aptr->index_order == 0))

	BuildIndex("idxordcl");

		// build non-clustered
index
		if (aptr->build_index ==
1)

	BuildIndex("idxordnc");
	}

}


//========================================
================================
//
// Function   : LoadNewOrderTable
//
//========================================
================================

void LoadNewOrderTable(LOADER_TIME_STRUCT
*new_order_time_start)
{
    int         i;
    long        o_id;
    short       o_d_id;
    short       o_w_id;
	RETCODE        rc;
	DBINT          rcint;

	// Bind NEW-ORDER data

    rc = bcp_bind(o_hdbc2, (BYTE *)
&o_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc2);

    rc = bcp_bind(o_hdbc2, (BYTE *)
&o_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc2);

    rc = bcp_bind(o_hdbc2, (BYTE *)
&o_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 3);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc2);

	for (i = first_new_order; i <
last_new_order; i++)
	{
		o_id    =
orders_buf[i].o_id;
		o_d_id  =
orders_buf[i].o_d_id;
		o_w_id  =
orders_buf[i].o_w_id;

		rc = bcp_sendrow(o_hdbc2);
		if (rc != SUCCEED)

	HandleErrorDBC(o_hdbc2);


		new_order_rows_loaded++;

		CheckForCommit(o_hdbc2,
o_hstmt2, new_order_rows_loaded,
"new_order", &new_order_time_start-
>time_start);
	}

	// rcint = bcp_batch(o_hdbc2);
	// if (rcint < 0)
	//     HandleErrorDBC(o_hdbc2);


	if ((o_w_id == aptr-
>num_warehouses) && (o_d_id == 10))
	{
		rcint = bcp_done(o_hdbc2);

		if (rcint < 0)

	HandleErrorDBC(o_hdbc2);


		SQLFreeStmt(o_hstmt2,
SQL_DROP);
		SQLDisconnect(o_hdbc2);
		SQLFreeConnect(o_hdbc2);

		// if build index after
load...
		if ((aptr->build_index ==
1) && (aptr->index_order == 0))

	BuildIndex("idxnodcl");

	}

}


//========================================
================================
//
// Function   : LoadOrderLineTable
//
//========================================
================================

void
LoadOrderLineTable(LOADER_TIME_STRUCT
*order_line_time_start)
{
	int         i,j;
    long        o_id;
	short          o_d_id;
	short          o_w_id;
    long        ol;
	long           ol_i_id;
    short       ol_supply_w_id;
    short       ol_quantity;
    double      ol_amount;
    char
ol_dist_info[DIST_INFO_LEN+1];
```

```
	char
	ol_delivery_d[OL_DELIVERY_D_LEN+1
];
	RETCODE		rc;
	DBINT		rcint;

	// bind ORDER-LINE data
    rc = bcp_bind(o_hdbc3, (BYTE *)
&o_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 1);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *)
&o_d_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 2);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *)
&o_w_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT2, 3);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *) &ol,
0, SQL_VARLEN_DATA, NULL, 0, SQLINT4, 4);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *)
&ol_i_id, 0, SQL_VARLEN_DATA, NULL, 0,
SQLINT4, 5);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *)
&ol_supply_w_id, 0, SQL_VARLEN_DATA,
NULL, 0, SQLINT2, 6);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

	rc = bcp_bind(o_hdbc3, (BYTE *)
&ol_delivery_d, 0, OL_DELIVERY_D_LEN,
NULL, 0, SQLCHARACTER, 7);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *)
&ol_quantity, 0, SQL_VARLEN_DATA, NULL,
0, SQLINT2, 8);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *)
&ol_amount, 0, SQL_VARLEN_DATA, NULL, 0,
SQLFLT8, 9);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

    rc = bcp_bind(o_hdbc3, (BYTE *)
ol_dist_info, 0, DIST_INFO_LEN, NULL, 0,
0, 10);
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

	for (i = 0; i <
orders_per_district; i++)
    {
		o_id	=
orders_buf[i].o_id;
		o_d_id  =
orders_buf[i].o_d_id;
		o_w_id  =
orders_buf[i].o_w_id;

		for (j=0; j <
orders_buf[i].o_ol_cnt; j++)
		{
			ol		=
orders_buf[i].o_ol[j].ol;
			ol_i_id		=
orders_buf[i].o_ol[j].ol_i_id;
			ol_supply_w_id =
orders_buf[i].o_ol[j].ol_supply_w_id;
			ol_quantity	=
orders_buf[i].o_ol[j].ol_quantity;
			ol_amount	=
orders_buf[i].o_ol[j].ol_amount;


	strcpy(ol_delivery_d,orders_buf[i
].o_ol[j].ol_delivery_d);


	strcpy(ol_dist_info,orders_buf[i]
.o_ol[j].ol_dist_info);

			rc =
bcp_sendrow(o_hdbc3);
			if (rc != SUCCEED)


	HandleErrorDBC(o_hdbc3);



	order_line_rows_loaded++;

	CheckForCommit(o_hdbc3, o_hstmt3,
order_line_rows_loaded, "order_line",
&order_line_time_start->time_start);
		}

	}

	// rcint = bcp_batch(o_hdbc3);
	// if (rcint < 0)
	//	HandleErrorDBC(o_hdbc3);


	if ((o_w_id == aptr-
>num_warehouses) && (o_d_id == 10))
	{
		rcint = bcp_done(o_hdbc3);

		if (rcint < 0)

	HandleErrorDBC(o_hdbc3);


		SQLFreeStmt(o_hstmt3,
SQL_DROP);
		SQLDisconnect(o_hdbc3);
		SQLFreeConnect(o_hdbc3);

		// if build index after
load...
		if ((aptr->build_index ==
1) && (aptr->index_order == 0))

	BuildIndex("idxodlcl");
```

```
        }

}


//======================================
================================
//
// Function   : GetPermutation
//
//======================================
================================

void GetPermutation(int perm[], int n)
{
        int i, r, t;

        for (i=1;i<=n;i++)
                perm[i] = i;

        for (i=1;i<=n;i++)
        {
                r = RandomNumber(i,n);
                t = perm[i];
                perm[i] = perm[r];
                perm[r] = t;
        }

}


//======================================
================================
//
// Function   : CheckForCommit
//
//======================================
================================

void CheckForCommit(HDBC hdbc,

        HSTMT hstmt,
                              int
rows_loaded,

        char *table_name,
                              long
*time_start)
{

    long       time_end, time_diff;
        // DBINT       rcint;

    if ( !(rows_loaded % aptr->batch) )
    {

                // rcint =
bcp_batch(hdbc);
                // if (rcint < 0)
                //
        HandleErrorDBC(hdbc);

                time_end = (TimeNow() /
MILLI);
                time_diff = time_end -
*time_start;

                printf("-> Loaded %ld rows
into %s in %ld sec - Total = %d (%.2f
rps)\n",
                       aptr->batch,
                       table_name,
                              time_diff,
                       rows_loaded,
                              (float)
aptr->batch / (time_diff ? time_diff :
1L));

                *time_start = time_end;
    }

    return;
}


//======================================
================================
//
// Function   : OpenConnections
//
//======================================
================================

void OpenConnections()
{

        RETCODE         rc;

        char
        szDriverString[300];
        char
        szDriverStringOut[1024];
        SQLSMALLINT
        cbDriverStringOut;



        SQLAllocHandle(SQL_HANDLE_ENV,
SQL_NULL_HANDLE, &henv );

        SQLSetEnvAttr(henv,
SQL_ATTR_ODBC_VERSION,
(void*)SQL_OV_ODBC3, 0 );

        SQLAllocHandle(SQL_HANDLE_DBC,
henv , &i_hdbc1);
        SQLAllocHandle(SQL_HANDLE_DBC,
henv , &w_hdbc1);
        SQLAllocHandle(SQL_HANDLE_DBC,
henv , &c_hdbc1);
        SQLAllocHandle(SQL_HANDLE_DBC,
henv , &c_hdbc2);
        SQLAllocHandle(SQL_HANDLE_DBC,
henv , &o_hdbc1);
        SQLAllocHandle(SQL_HANDLE_DBC,
henv , &o_hdbc2);
        SQLAllocHandle(SQL_HANDLE_DBC,
henv , &o_hdbc3);

        SQLSetConnectAttr(i_hdbc1,
SQL_COPT_SS_BCP, (void *)SQL_BCP_ON,
SQL_IS_INTEGER );
        SQLSetConnectAttr(w_hdbc1,
SQL_COPT_SS_BCP, (void *)SQL_BCP_ON,
SQL_IS_INTEGER );
        SQLSetConnectAttr(c_hdbc1,
SQL_COPT_SS_BCP, (void *)SQL_BCP_ON,
SQL_IS_INTEGER );
        SQLSetConnectAttr(c_hdbc2,
SQL_COPT_SS_BCP, (void *)SQL_BCP_ON,
SQL_IS_INTEGER );
```

```
	SQLSetConnectAttr(o_hdbc1,
SQL_COPT_SS_BCP, (void *)SQL_BCP_ON,
SQL_IS_INTEGER );
	SQLSetConnectAttr(o_hdbc2,
SQL_COPT_SS_BCP, (void *)SQL_BCP_ON,
SQL_IS_INTEGER );
	SQLSetConnectAttr(o_hdbc3,
SQL_COPT_SS_BCP, (void *)SQL_BCP_ON,
SQL_IS_INTEGER );

	// Open connections to SQL Server

	// Connection 1

	sprintf( szDriverString ,
"DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=
%s" ,

	  aptr->server,

	  aptr->user,

	  aptr->password,

	  aptr->database );

	rc = SQLSetConnectOption
(i_hdbc1, SQL_PACKET_SIZE, aptr-
>pack_size);
	if (rc != SUCCEED)
		HandleErrorDBC(i_hdbc1);

	rc = SQLDriverConnect ( i_hdbc1,

NULL,

	(SQLCHAR*)&szDriverString[0] ,

	SQL_NTS,

	(SQLCHAR*)&szDriverStringOut[0],

	sizeof(szDriverStringOut),

	&cbDriverStringOut,

	SQL_DRIVER_NOPROMPT );
	if (rc != SUCCEED)
		HandleErrorDBC(i_hdbc1);

	// Connection 2

	sprintf( szDriverString ,
"DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=
%s" ,

	  aptr->server,

	  aptr->user,

	  aptr->password,

	  aptr->database );

	rc = SQLSetConnectOption
(w_hdbc1, SQL_PACKET_SIZE, aptr-
>pack_size);
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	rc = SQLDriverConnect ( w_hdbc1,

		NULL,

	(SQLCHAR*)&szDriverString[0] ,

		SQL_NTS,

	(SQLCHAR*)&szDriverStringOut[0],

		sizeof(szDriverStringOut),

		&cbDriverStringOut,

		SQL_DRIVER_NOPROMPT );
	if (rc != SUCCEED)
		HandleErrorDBC(w_hdbc1);

	// Connection 3

	sprintf( szDriverString ,
"DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=
%s" ,

	  aptr->server,

	  aptr->user,

	  aptr->password,

	  aptr->database );

	rc = SQLSetConnectOption
(c_hdbc1, SQL_PACKET_SIZE, aptr-
>pack_size);
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

	rc = SQLDriverConnect ( c_hdbc1,

		NULL,

	(SQLCHAR*)&szDriverString[0] ,

		SQL_NTS,

	(SQLCHAR*)&szDriverStringOut[0],

		sizeof(szDriverStringOut),

		&cbDriverStringOut,

		SQL_DRIVER_NOPROMPT );
	if (rc != SUCCEED)
		HandleErrorDBC(c_hdbc1);

	// Connection 4

	sprintf( szDriverString ,
"DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=
%s" ,

	  aptr->server,
```

```
          aptr->user,

          aptr->password,

          aptr->database );

        rc = SQLSetConnectOption
(c_hdbc2, SQL_PACKET_SIZE, aptr-
>pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

        rc = SQLDriverConnect ( c_hdbc2,

                NULL,


        (SQLCHAR*)&szDriverString[0] ,

                SQL_NTS,


        (SQLCHAR*)&szDriverStringOut[0],

                sizeof(szDriverStringOut),

                &cbDriverStringOut,

                SQL_DRIVER_NOPROMPT );
        if (rc != SUCCEED)
                HandleErrorDBC(c_hdbc2);

        // Connection 5

        sprintf( szDriverString ,
"DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=
%s" ,

          aptr->server,

          aptr->user,

          aptr->password,

          aptr->database );

        rc = SQLSetConnectOption
(o_hdbc1, SQL_PACKET_SIZE, aptr-
>pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

        rc = SQLDriverConnect ( o_hdbc1,

                NULL,


        (SQLCHAR*)&szDriverString[0] ,

                SQL_NTS,


        (SQLCHAR*)&szDriverStringOut[0],

                sizeof(szDriverStringOut),

                &cbDriverStringOut,

                SQL_DRIVER_NOPROMPT );
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc1);

        // Connection 6

        sprintf( szDriverString ,
"DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=
%s" ,

          aptr->server,

          aptr->user,

          aptr->password,

          aptr->database );

        rc = SQLSetConnectOption
(o_hdbc2, SQL_PACKET_SIZE, aptr-
>pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc2);

        rc = SQLDriverConnect ( o_hdbc2,

                NULL,


        (SQLCHAR*)&szDriverString[0] ,

                SQL_NTS,


        (SQLCHAR*)&szDriverStringOut[0],

                sizeof(szDriverStringOut),

                &cbDriverStringOut,

                SQL_DRIVER_NOPROMPT );
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc2);

        // Connection 7

        sprintf( szDriverString ,
"DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=
%s" ,

          aptr->server,

          aptr->user,

          aptr->password,

          aptr->database );

        rc = SQLSetConnectOption
(o_hdbc3, SQL_PACKET_SIZE, aptr-
>pack_size);
        if (rc != SUCCEED)
                HandleErrorDBC(o_hdbc3);

        rc = SQLDriverConnect ( o_hdbc3,

                NULL,


        (SQLCHAR*)&szDriverString[0] ,
```

```
		SQL_NTS,


	(SQLCHAR*)&szDriverStringOut[0],

		sizeof(szDriverStringOut),

		&cbDriverStringOut,

		SQL_DRIVER_NOPROMPT );
	if (rc != SUCCEED)
		HandleErrorDBC(o_hdbc3);

}

//========================================
===============================
//
// Function name: BuildIndex
//
//========================================
===============================

void BuildIndex(char   *index_script)
{
	char   cmd[256];

	printf("Starting index creation:
%s\n",index_script);

	sprintf(cmd, "osql -S%s -U%s -P%s
-e -i%s\\%s.sql > %s%s.log",
				aptr-
>server,
				aptr->user,
				aptr-
>password,
				aptr-
>index_script_path,

	index_script,
				aptr-
>log_path,

	index_script);

	system(cmd);

	printf("Finished index creation:
%s\n",index_script);
}


void HandleErrorDBC (SQLHDBC  hdbc1)
{

	SQLCHAR		SqlState[6],
Msg[SQL_MAX_MESSAGE_LENGTH];
	SQLINTEGER  NativeError;
	SQLSMALLINT i, MsgLen;
	SQLRETURN   rc2;
	char		timebuf[128];
	char		datebuf[128];
	char		err_log_path[256];
	FILE		*fp1;

	i = 1;
	while (( rc2 =
SQLGetDiagRec(SQL_HANDLE_DBC , hdbc1, i,
SqlState , &NativeError,
						Msg,
sizeof(Msg) , &MsgLen )) != SQL_NO_DATA )
	{

		sprintf( szLastError ,
"%s" ,  Msg );

		_strtime(timebuf);
		_strdate(datebuf);

		printf( "[%s : %s] %s\n" ,
datebuf, timebuf, szLastError);

		strcpy(err_log_path,aptr-
>log_path);

	strcat(err_log_path,"tpccldr.err"
);
		fp1 =
fopen(err_log_path,"w");
		//fp1 =
fopen("logs\\tpccldr.err","w");
		if (fp1 == NULL)

			printf("ERROR:
Unable to open errorlog file.\n");
		else
		{
			fprintf(fp1, "[%s
: %s] %s\n" , datebuf, timebuf,
szLastError);
			fclose(fp1);
		}

		i++;

	}
}

void HandleErrorSTMT (HSTMT  hstmt1)
{

	SQLCHAR		SqlState[6],
Msg[SQL_MAX_MESSAGE_LENGTH];
	SQLINTEGER  NativeError;
	SQLSMALLINT i, MsgLen;
	SQLRETURN   rc2;
	char		timebuf[128];
	char		datebuf[128];
	char		err_log_path[256];
	FILE		*fp1;

	i = 1;
	while (( rc2 =
SQLGetDiagRec(SQL_HANDLE_STMT , hstmt1,
i, SqlState , &NativeError,
						Msg,
sizeof(Msg) , &MsgLen )) != SQL_NO_DATA )
	{

		sprintf( szLastError ,
"%s" ,  Msg );

		_strtime(timebuf);
		_strdate(datebuf);

		printf( "[%s : %s] %s\n" ,
datebuf, timebuf, szLastError);

		strcpy(err_log_path,aptr-
>log_path);
```

```
	strcat(err_log_path,"tpccldr.err"
);
		fp1 =
fopen(err_log_path,"w");
		//fp1 =
fopen("logs\\tpccldr.err","w");
		if (fp1 == NULL)

			printf("ERROR:
Unable to open errorlog file.\n");
		else
		{
			fprintf(fp1, "[%s
: %s] %s\n" , datebuf, timebuf,
szLastError);
			fclose(fp1);
		}

		i++;

	}
}


void FormatDate ( char* szTimeCOutput )
{

	struct tm when;
	time_t now;

	time( &now );
	when  = *localtime( &now );

	mktime( &when );

	// odbc datetime format
	strftime( szTimeCOutput , 30 ,
"%Y-%m-%d %H:%M:%S.000", &when );

	return;

}

//=======================================
===============================
//
// Function   : CheckDataBase
//
//=======================================
===============================

void CheckDataBase()
{

	RETCODE		rc;

	char
	szDriverString[300];
	char
	szDriverStringOut[1024];
	char
	TablesBitMap[9] = {"000000000"};
	int				i,
ExitFlag;

	SQLSMALLINT
	cbDriverStringOut;
	SQLCHAR
	TabName[10];
	SQLINTEGER		TabNameInd,
TabCount, TabCountInd;


	ExitFlag = 0;

	SQLAllocHandle(SQL_HANDLE_ENV,
SQL_NULL_HANDLE, &henv );

	SQLSetEnvAttr(henv,
SQL_ATTR_ODBC_VERSION,
(void*)SQL_OV_ODBC3, 0 );

	SQLAllocHandle(SQL_HANDLE_DBC,
henv , &v_hdbc);

	SQLSetConnectAttr(v_hdbc,
SQL_COPT_SS_BCP, (void *)SQL_BCP_ON,
SQL_IS_INTEGER );

	// Open connection to SQL Server

	sprintf( szDriverString ,
"DRIVER={SQL
Server};SERVER=%s;UID=%s;PWD=%s;DATABASE=
%s" ,

	  aptr->server,

	  aptr->user,

	  aptr->password,

	  aptr->database );

	rc = SQLSetConnectAttr( v_hdbc,
SQL_ATTR_PACKET_SIZE, (SQLPOINTER)aptr-
>pack_size, SQL_IS_UINTEGER );
	if (rc != SQL_SUCCESS)
		HandleErrorDBC(v_hdbc);

	rc = SQLDriverConnect ( v_hdbc,

NULL,

	(SQLCHAR*)&szDriverString[0] ,

	SQL_NTS,

	(SQLCHAR*)&szDriverStringOut[0],

	sizeof(szDriverStringOut),

	&cbDriverStringOut,

	SQL_DRIVER_NOPROMPT );

	// if the rc is SQL_ERROR, the
the TPCC database probably does not exist
	if (rc == SQL_ERROR)
	{
		printf("The database TPCC
does not appear to exist!\n");
		printf("\nCheck LOGS\\
directory for database creation
errors.\n");

		// cleanup database
connections and handles
```

```
		SQLFreeHandle(SQL_HANDLE_STMT, v_hstmt);
		SQLDisconnect(v_hdbc);

		SQLFreeHandle(SQL_HANDLE_DBC, v_hdbc);

		// since there is not a database, exit back to SETUP.CMD
		exit(1);
	}

	if ( SQLAllocHandle(SQL_HANDLE_STMT, v_hdbc , &v_hstmt) != SQL_SUCCESS )
		HandleErrorDBC(v_hdbc);

	if ( SQLBindCol(v_hstmt, 1, SQL_C_ULONG, &TabCount, 0, &TabCountInd) != SQL_SUCCESS )
		HandleErrorSTMT(v_hstmt);

	// count the number of user tables from sysobjects
	rc = SQLExecDirect(v_hstmt, "select count(*) from sysobjects where xtype = \'U\'", SQL_NTS);
	if ((rc != SQL_SUCCESS) && (rc != SQL_SUCCESS_WITH_INFO))
		HandleErrorSTMT(v_hstmt);


	if ( SQLFetch(v_hstmt) != SQL_SUCCESS )
		HandleErrorSTMT(v_hstmt);

	// if the number of tables is less than 9, select all the user tables in TPCC
	if (TabCount != 9)
	{

	SQLFreeHandle(SQL_HANDLE_STMT, v_hstmt);


	SQLAllocHandle(SQL_HANDLE_STMT, v_hdbc , &v_hstmt);

		if ( SQLBindCol(v_hstmt, 1, SQL_C_CHAR, &TabName, sizeof(TabName), &TabNameInd) != SQL_SUCCESS )

	HandleErrorSTMT(v_hstmt);

		// select the list of user tables into a result set
		rc = SQLExecDirect(v_hstmt, "select * from sysobjects where xtype = \'U\'", SQL_NTS);
		if ((rc != SQL_SUCCESS) && (rc != SQL_SUCCESS_WITH_INFO))

	HandleErrorSTMT(v_hstmt);

		// go through the result set and set the bitmap for each found table
		// set the bitmap to '1' if the table name is found

		while ((rc = SQLFetch(v_hstmt)) != SQL_NO_DATA)
		{
			switch( TabName[0] )
			{
			case 'w':

	TablesBitMap[0] = '1';
				break;
			case 'd':

	TablesBitMap[1] = '1';
				break;
			case 'c':

	TablesBitMap[2] = '1';
				break;
			case 'h':

	TablesBitMap[3] = '1';
				break;
			case 'n':

	TablesBitMap[4] = '1';
				break;
			case 'o':
				if (TabName[5] = 's')

	TablesBitMap[5] = '1';
				if (TabName[5] = '_')

	TablesBitMap[6] = '1';
				break;
			case 'i':

	TablesBitMap[7] = '1';
				break;
			case 's':

	TablesBitMap[8] = '1';
				break;
			}
		}

		// a '0' ExitFlag means do NOT exit the loader early, a '1' means exit the loader early
		ExitFlag = 0;

		// interate through the bitmap to display which table(s) is actually missing
		for (i = 0; i <= 8; i++)
		{
			switch(i)
			{
			case 0:
				if (TablesBitMap[i] == '0')

				{

	printf("The Warehouse table is missing or damaged.\n");
```

```
	ExitFlag = 1;
					}
					break;
				case 1:
					if
(TablesBitMap[i] == '0')
					{

	printf("The District table is
missing or damaged.\n");

	ExitFlag = 1;
					}
					break;
				case 2:
					if
(TablesBitMap[i] == '0')
					{

	printf("The Customer table is
missing or damaged.\n");

	ExitFlag = 1;
					}
					break;
				case 3:
					if
(TablesBitMap[i] == '0')
					{

	printf("The History table is
missing or damaged.\n");

	ExitFlag = 1;
					}
					break;
				case 4:
					if
(TablesBitMap[i] == '0')
					{

	printf("The New_Order table is
missing or damaged.\n");

	ExitFlag = 1;
					}
					break;
				case 5:
					if
(TablesBitMap[i] == '0')
					{


	printf("The Orders table is
missing or damaged.\n");

	ExitFlag = 1;
					}
					break;
				case 6:
					if
(TablesBitMap[i] == '0')
					{

	printf("The Order_Line table is
missing or damaged.\n");

	ExitFlag = 1;
					}
					break;
				case 7:
					if
(TablesBitMap[i] == '0')
					{

	printf("The Item table is missing
or damaged.\n");

	ExitFlag = 1;
					}
					break;
				case 8:
					if
(TablesBitMap[i] == '0')
					{

	printf("The Stock table is
missing or damaged.\n");

	ExitFlag = 1;
					}
					break;
				}
			}

			// if one or more tables
are missing, display message and exit the
loader
			if (ExitFlag = 1)
			{
				printf("\nExiting
TPC-C Loader!\n");
				printf("\nCheck
LOGS\\ directory for database\n");
				printf("or table
creation errors.\n");

				// cleanup
database connections and handles

	SQLFreeHandle(SQL_HANDLE_STMT,
v_hstmt);

	SQLDisconnect(v_hdbc);

	SQLFreeHandle(SQL_HANDLE_DBC,
v_hdbc);

				exit(1);
			}
	}

	// cleanup database connections
and handles
	SQLFreeHandle(SQL_HANDLE_STMT,
v_hstmt);
	SQLDisconnect(v_hdbc);
	SQLFreeHandle(SQL_HANDLE_DBC,
v_hdbc);

	return;
}
```

Random.c

```
//	File:		RANDOM.C
//				Microsoft
TPC-C Kit Ver. 4.41
```

```
//                              Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000,
2001
//      Purpose:       Random number
generation routines for database loader


// Includes
#include "tpcc.h"
#include "math.h"

// Defines
#define A           16807
#define M       2147483647
#define Q           127773      /* M div
A */
#define R             2836      /* M mod
A */
#define Thread         __declspec(thread)

// Globals
long   Thread Seed = 0;       /* thread
local seed */

/****************************************
*************************************
*
*
* random -
*
*       Implements a GOOD pseudo random
number generator.  This generator     *
*       will/should? run the complete
period before repeating.               *
*
*
* Copied from:
*
*       Random Numbers Generators: Good
Ones Are Hard to Find.                *
*       Communications of the ACM -
October 1988 Volume 31 Number 10
*
*
*
* Machine Dependencies:
*
*       long must be 2 ^ 31 - 1 or
greater.
*
*
*
*****************************************
************************************/

/****************************************
*************************************
* seed - load the Seed value used in
irand and drand.  Should be used before
*
*       first call to irand or drand.
*
*****************************************
************************************/

void seed(long val)
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering
seed()...\n", (int)
GetCurrentThreadId());
        printf("Old Seed %ld New Seed
%ld\n",Seed, val);
#endif

    if ( val < 0 )
            val = abs(val);

    Seed = val;

}


/****************************************
*************************************
*
*
* irand - returns a 32 bit integer pseudo
random number with a period of     *
*       1 to 2 ^ 32 - 1.
*
*
*
* parameters:
*
*       none.
*
*
*
* returns:
*
*       32 bit integer - defined as long
( see above ).                        *
*
*
* side effects:
*
*       seed get recomputed.
*
*****************************************
***********************************/

long irand()
{
    register long    s;      /* copy of
seed */
    register long    test;   /* test flag
*/
    register long    hi;     /* tmp value
for speed */
    register long    lo;     /* tmp value
for speed */

#ifdef DEBUG

    printf("[%ld]DBG: Entering
irand()...\n", (int)
GetCurrentThreadId());
#endif

    s = Seed;
    hi = s / Q;
    lo = s % Q;

    test = A * lo - R * hi;
    if ( test > 0 )
            Seed = test;
    else
            Seed = test + M;
```

```
    return( Seed );

}

/***************************************
***********************************
*
*
* drand - returns a double pseudo random
number between 0.0 and 1.0.         *
*       See irand.
*
****************************************
**********************************/
double drand()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering
drand()...\n", (int)
GetCurrentThreadId());
#endif

    return( (double)irand() /
2147483647.0);
}


//=====================================
================================
// Function   : RandomNumber
//
// Description:
//=====================================
================================
long RandomNumber(long lower, long upper)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering
RandomNumber()...\n", (int)
GetCurrentThreadId());
#endif

       if ( upper == lower )  /* pgd 08-
13-96 perf enhancement */
              return lower;

       upper++;

    if ( upper <= lower )
              rand_num = upper;
    else
              rand_num = lower + irand()
% (upper - lower); /* pgd 08-13-96 perf
enhancement */


#ifdef DEBUG
    printf("[%ld]DBG: RandomNumber
between %ld & %ld ==> %ld\n",
                                (int)
GetCurrentThreadId(), lower, upper,
rand_num);
#endif


    return rand_num;
}


#if 0

//Orginal code pgd 08/13/96

long RandomNumber(long lower,
                                long
upper)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering
RandomNumber()...\n", (int)
GetCurrentThreadId());
#endif

       upper++;

    if ((upper <= lower))
              rand_num = upper;
    else
              rand_num = lower + irand()
% ((upper > lower) ? upper - lower :
upper);


#ifdef DEBUG
    printf("[%ld]DBG: RandomNumber
between %ld & %ld ==> %ld\n",
                              (int)
GetCurrentThreadId(), lower, upper,
rand_num);
#endif


    return rand_num;
}
#endif

//=====================================
================================
// Function   : NURand
//
// Description:
//=====================================
================================
long NURand(int iConst,
           long x,
           long y,
           long C)
{
    long rand_num;

#ifdef DEBUG
    printf("[%ld]DBG: Entering
NURand()...\n", (int)
GetCurrentThreadId());
#endif

    rand_num = (((RandomNumber(0,iConst)
| RandomNumber(x,y)) + C) % (y-x+1))+x;

#ifdef DEBUG
    printf("[%ld]DBG: NURand: num =
%d\n", (int) GetCurrentThreadId(),
rand_num);
#endif
```

```
    return rand_num;
}
```

Strings.c

```
//      File:           STRINGS.C
//                              Microsoft
TPC-C Kit Ver. 4.41
//                              Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000,
2001
//      Purpose:        Source file for
database loader string functions


// Includes
#include "tpcc.h"
#include <string.h>
#include <ctype.h>

//=======================================
================================
//
// Function name: MakeAddress
//
//=======================================
================================

void MakeAddress(char *street_1,
                              char
*street_2,
                              char
*city,
                              char
*state,
                              char *zip)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering
MakeAddress()\n", (int)
GetCurrentThreadId());
#endif

    MakeAlphaString (10, 20, ADDRESS_LEN,
street_1);
    MakeAlphaString (10, 20, ADDRESS_LEN,
street_2);
    MakeAlphaString (10, 20, ADDRESS_LEN,
city);
    MakeAlphaString ( 2,  2, STATE_LEN,
state);
    MakeZipNumberString( 9,  9, ZIP_LEN,
zip);

#ifdef DEBUG
    printf("[%ld]DBG: MakeAddress:
street_1: %s, street_2: %s, city: %s,
state: %s, zip: %s\n",
                        (int)
GetCurrentThreadId(), street_1, street_2,
city, state, zip);
#endif

    return;

}

//=======================================
================================
//
// Function name: LastName
//
//=======================================
================================

void LastName(int  num,
                     char *name)
{
    static char *n[] =
    {
              "BAR" , "OUGHT", "ABLE" ,
"PRI"  , "PRES",
              "ESE" , "ANTI" , "CALLY",
"ATION", "EING"
    };

#ifdef DEBUG
    printf("[%ld]DBG: Entering
LastName()\n", (int)
GetCurrentThreadId());
#endif

    if ((num >= 0) && (num < 1000))
    {
              strcpy(name,
n[(num/100)%10]);
              strcat(name,
n[(num/10)%10]);
              strcat(name,
n[(num/1)%10]);

              if (strlen(name) <
LAST_NAME_LEN)
              {

       PaddString(LAST_NAME_LEN, name);
              }

    }
    else
    {
              printf("\nError in
LastName()... num <%ld> out of range
(0,999)\n", num);
              exit(-1);
    }


#ifdef DEBUG
    printf("[%ld]DBG: LastName: num =
[%d] ==> [%d][%d][%d]\n",
                      (int)
GetCurrentThreadId(), num, num/100,
(num/10)%10, num%10);
       printf("[%ld]DBG: LastName:
String = %s\n", (int)
GetCurrentThreadId(), name);
#endif

    return;

}


//=======================================
================================
//
// Function name: MakeAlphaString
```

```
//
//========================================
================================

//philipdu 08/13/96 Changed
MakeAlphaString to use A-Z, a-z, and 0-9
in
//accordance with spec see below:
//The spec says:
//4.3.2.2      The notation random a-
string [x .. y]
//(respectively, n-string [x .. y])
represents a string of random
alphanumeric
//(respectively, numeric) characters of a
random length of minimum x, maximum y,
//and mean (y+x)/2. Alphanumerics are
A..Z, a..z, and 0..9.  The only other
//requirement is that the character set
used "must be able to represent a minimum
//of 128 different characters".  We are
using 8-bit chars, so this is a non
issue.
//It is completely unreasonable to stuff
non-printing chars into the text fields.
//-CLevine 08/13/96

int MakeAlphaString( int  x, int  y, int
z, char *str)
{
       int            len;
       int            i;
    char    cc = 'a';
       static  char chArray[] =
"0123456789ABCDEFGHIJKLMNOPQRSTUVWXYZabcd
efghijklmnopqrstuvwxyz";
       static  int     chArrayMax = 61;

#ifdef DEBUG
    printf("[%ld]DBG: Entering
MakeAlphaString()\n", (int)
GetCurrentThreadId());
#endif

       len= RandomNumber(x, y);

       for (i=0; i<len; i++)
       {
              cc =
chArray[RandomNumber(0, chArrayMax)];
        str[i] = cc;
    }
       //if ( len < z )
       //     memset(str+len, ' ', z -
len);
       str[len] = 0;

    return len;
}


//========================================
================================
//
// Function name: MakeOriginalAlphaString
//
//========================================
================================

int MakeOriginalAlphaString(int  x,
              int  y,

                 int  z,

              char *str,

          int  percent)
{
    int        len;
    int        val;
    int        start;

#ifdef DEBUG
    printf("[%ld]DBG: Entering
MakeOriginalAlphaString()\n", (int)
GetCurrentThreadId());
#endif

    // verify prercentage is valid
    if ((percent < 0) || (percent > 100))
    {

        printf("MakeOrigianlAlphaString:
Invalid percentage: %d\n", percent);
              exit(-1);
    }

    // verify string is at least 8 chars
in length
    if ((x + y) <= 8)
    {

        printf("MakeOriginalAlphaString:
string length must be >= 8\n");
              exit(-1);
    }

    // Make Alpha String
    len = MakeAlphaString(x,y, z, str);

    val = RandomNumber(1,100);
    if (val <= percent)
    {
              start = RandomNumber(0,
len - 8);
              strncpy(str + start,
"ORIGINAL", 8);
    }


#ifdef DEBUG
    printf("[%ld]DBG:
MakeOriginalAlphaString: : %s\n",
                        (int)
GetCurrentThreadId(), str);
#endif

    return strlen(str);
}


//========================================
================================
//
// Function name: MakeNumberString
//
//========================================
================================
int MakeNumberString(int  x, int  y, int
z, char *str)
```

```
{
        char tmp[16];

        //MakeNumberString is always
called MakeZipNumberString(16, 16, 16,
string)

        memset(str, '0', 16);
        itoa(RandomNumber(0, 99999999),
tmp, 10);
        memcpy(str, tmp, strlen(tmp));

        itoa(RandomNumber(0, 99999999),
tmp, 10);
        memcpy(str+8, tmp, strlen(tmp));

        str[16] = 0;

    return 16;
}


//========================================
================================
//
// Function name: MakeZipNumberString
//
//========================================
================================
int MakeZipNumberString(int  x, int  y,
int  z, char *str)
{
        char tmp[16];

        //MakeZipNumberString is always
called MakeZipNumberString(9, 9, 9,
string)

        strcpy(str, "000011111");

        itoa(RandomNumber(0, 9999), tmp,
10);
        memcpy(str, tmp, strlen(tmp));

    return 9;
}


//========================================
================================
//
// Function name: InitString
//
//========================================
================================
void InitString(char *str, int len)
{
#ifdef DEBUG
    printf("[%ld]DBG: Entering
InitString()\n", (int)
GetCurrentThreadId());
#endif

        memset(str, ' ', len);
    str[len] = 0;
}


//========================================
================================
// Function name: InitAddress
//
// Description:
//
//========================================
================================

void InitAddress(char *street_1, char
*street_2, char *city, char *state, char
*zip)
{

        memset(street_1, ' ',
ADDRESS_LEN+1);
        memset(street_2, ' ',
ADDRESS_LEN+1);
        memset(city, ' ', ADDRESS_LEN+1);

    street_1[ADDRESS_LEN+1] = 0;
    street_2[ADDRESS_LEN+1] = 0;
    city[ADDRESS_LEN+1]     = 0;

        memset(state, ' ', STATE_LEN+1);
    state[STATE_LEN+1] = 0;

        memset(zip, ' ', ZIP_LEN+1);
    zip[ZIP_LEN+1] = 0;
}


//========================================
================================
//
// Function name: PaddString
//
//========================================
================================

void PaddString(int max, char *name)
{
        int             len;

        len = strlen(name);
        if ( len < max )
                memset(name+len, ' ', max
- len);
        name[max] = 0;

        return;
}
```

Time.c

```
//      File:           TIME.C
//                              Microsoft
TPC-C Kit Ver. 4.41
//                              Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000,
2001
//      Purpose:        Source file for
time functions


// Includes
#include "tpcc.h"

// Globals
static long start_sec;
```

```
//=======================================
===============================
//
// Function name: TimeNow
//
//=======================================
===============================

long TimeNow()
{
   long        time_now;
   struct      _timeb el_time;

#ifdef DEBUG
    printf("[%ld]DBG: Entering
TimeNow()\n", (int)
GetCurrentThreadId());
#endif

   _ftime(&el_time);

   time_now = ((el_time.time - start_sec)
* 1000) + el_time.millitm;

   return time_now;
}
```

Tpcc.h

```
//     File:          TPCC.H
//                              Microsoft
TPC-C Kit Ver. 4.41
//                              Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000,
2001
//     Purpose:       Header file for
TPC-C database loader


// Build number of TPC Benchmark Kit
#define TPCKIT_VER    "4.41"

// General headers
#include <windows.h>
#include <winbase.h>
#include <stdlib.h>
#include <stdio.h>
#include <process.h>
#include <stddef.h>
#include <stdarg.h>
#include <string.h>
#include <time.h>
#include <sys\timeb.h>
#include <sys\types.h>

// ODBC headers
#include <sql.h>
#include <sqlext.h>
#include <odbcss.h>

// General constants
#define MILLI
1000
#define FALSE
       0
#define TRUE
       1
#define UNDEF
           -1
#define MINPRINTASCII
32
#define MAXPRINTASCII
126

// Default environment constants
#define SERVER
       ""
#define DATABASE
       "tpcc"
#define USER
       "sa"
#define PASSWORD
""

// Default loader arguments
#define BATCH
       10000
#define DEFLDPACKSIZE
       32768
#define LOADER_RES_FILE
"C:\\MSTPCC.440\\SETUP\\logs\\load.out"
#define LOG_PATH
"C:\\MSTPCC.440\\SETUP\\LOGS\\";
#define LOADER_NURAND_C
         123
#define DEF_STARTING_WAREHOUSE
       1
#define BUILD_INDEX
              1      // build both data
and indexes
#define INDEX_ORDER
              1      // build indexes
before load
#define SCALE_DOWN
              0   // build a normal
scale database
#define INDEX_SCRIPT_PATH     "scripts"

typedef struct
{
    char
       *server;
    char
       *database;
    char
       *user;
    char
       *password;
       BOOL
       tables_all;                  //
set if loading all tables
       BOOL
       table_item;                  //
set if loading ITEM table specifically
       BOOL
       table_warehouse;       // set if
loading WAREHOUSE, DISTRICT, and STOCK
       BOOL
       table_customer;              //
set if loading CUSTOMER and HISTORY
       BOOL
       table_orders;          // set if
loading NEW-ORDER, ORDERS, ORDER-LINE
    long
       num_warehouses;
```

```
    long
       batch;
    long
       verbose;
       long
       pack_size;
       char
       *loader_res_file;
       char
       *log_path;
       char
       *synch_servername;
       long
       case_sensitivity;
       long
       starting_warehouse;
       long
       build_index;
       long
       index_order;
       long
       scale_down;
       char
       *index_script_path;
} TPCCLDR_ARGS;

// String length constants
       #define SERVER_NAME_LEN
20
       #define DATABASE_NAME_LEN
20
       #define USER_NAME_LEN
20
       #define PASSWORD_LEN
20
       #define TABLE_NAME_LEN
20
       #define I_DATA_LEN
50
       #define I_NAME_LEN
24
       #define BRAND_LEN
1
       #define LAST_NAME_LEN
16
       #define W_NAME_LEN
10
       #define ADDRESS_LEN
20
       #define STATE_LEN
2
       #define ZIP_LEN
9
       #define S_DIST_LEN
24
       #define S_DATA_LEN
50
       #define D_NAME_LEN
10
       #define FIRST_NAME_LEN
16
       #define MIDDLE_NAME_LEN
2
       #define PHONE_LEN
16
       #define CREDIT_LEN
2
       #define C_DATA_LEN
500
       #define H_DATA_LEN
24
       #define DIST_INFO_LEN
24
       #define MAX_OL_NEW_ORDER_ITEMS
15
       #define MAX_OL_ORDER_STATUS_ITEMS
15
       #define STATUS_LEN
25
       #define OL_DIST_INFO_LEN
          24
       #define C_SINCE_LEN
                 23
       #define H_DATE_LEN
                 23
       #define OL_DELIVERY_D_LEN
          23
       #define O_ENTRY_D_LEN
23


// Functions in random.c
void   seed();
long   irand();
double drand();
void   WUCreate();
short  WURand();
long   RandomNumber(long lower, long
upper);

// Functions in getargs.c;
void   GetArgsLoader();
void   GetArgsLoaderUsage();

// Functions in time.c
long   TimeNow();

// Functions in strings.c
void   MakeAddress();
void   LastName();
int    MakeAlphaString();
int    MakeOriginalAlphaString();
int    MakeNumberString();
int    MakeZipNumberString();
void   InitString();
void   InitAddress();
void   PaddString();
```

Getargs.c

```
//     File:         GETARGS.C
//                          Microsoft
TPC-C Kit Ver. 4.41
//                          Copyright
Microsoft, 1996, 1997, 1998, 1999, 2000,
2001
//     Purpose:      Source file for
command line processing


// Includes
#include "tpcc.h"

//=======================================
================================
//
// Function name: GetArgsLoader
//
//=======================================
================================
```

```
void GetArgsLoader(int argc, char **argv,
TPCCLDR_ARGS *pargs)
{
    int         i;
    char        *ptr;

#ifdef DEBUG
    printf("[%ld]DBG: Entering
GetArgsLoader()\n", (int)
GetCurrentThreadId());
#endif

    /* init args struct with some useful
values */
    pargs->server                   =
SERVER;
    pargs->user                     =
USER;
    pargs->password                 =
PASSWORD;
    pargs->database                 =
DATABASE;
    pargs->batch                    =
BATCH;
    pargs->num_warehouses           =
UNDEF;
        pargs->tables_all
        =  TRUE;
        pargs->table_item
        =  FALSE;
        pargs->table_warehouse      =
FALSE;
        pargs->table_customer       =
FALSE;
        pargs->table_orders
        =  FALSE;
        pargs->loader_res_file
        =  LOADER_RES_FILE;
        pargs->log_path
                =  LOG_PATH;
        pargs->pack_size
        =  DEFLDPACKSIZE;
        pargs->starting_warehouse   =
DEF_STARTING_WAREHOUSE;
        pargs->build_index
        =  BUILD_INDEX;
        pargs->index_order
        =  INDEX_ORDER;
        pargs->index_script_path    =
INDEX_SCRIPT_PATH;
        pargs->scale_down
        =  SCALE_DOWN;

    /* check for zero command line args
*/
    if ( argc == 1 )
               GetArgsLoaderUsage();

    for (i = 1; i < argc; ++i)
    {
               if (argv[i][0] != '-' &&
argv[i][0] != '/')
               {
               printf("\nUnrecognized
command");
               GetArgsLoaderUsage();
               exit(1);
               }

               ptr = argv[i];

               switch (ptr[1])
               {
               case '?':     /* Fall
throught */

       GetArgsLoaderUsage();
                              break;

               case 'D':
                              pargs-
>database = ptr+2;
                              break;

               case 'P':
                              pargs-
>password = ptr+2;
                              break;

               case 'S':
                              pargs-
>server = ptr+2;
                              break;

               case 'U':
                              pargs->user
= ptr+2;
                              break;

               case 'b':
                              pargs-
>batch = atol(ptr+2);
                              break;

               case 'W':
                              pargs-
>num_warehouses = atol(ptr+2);
                              break;

               case 's':
                              pargs-
>starting_warehouse = atol(ptr+2);
                              break;

               case 't':
                              {

       pargs->tables_all = FALSE;
                                     if
(strcmp(ptr+2,"item") == 0)

       pargs->table_item = TRUE;

       else if
(strcmp(ptr+2,"warehouse") == 0)

       pargs->table_warehouse = TRUE;

       else if (strcmp(ptr+2,"customer")
== 0)

       pargs->table_customer = TRUE;

       else if (strcmp(ptr+2,"orders")
== 0)

       pargs->table_orders = TRUE;

       else
                                     {
```

```
	printf("\nUnrecognized command");

	GetArgsLoaderUsage();

	exit(1);
				}


	break;
				}

			case 'f':
				pargs-
>loader_res_file = ptr+2;
				break;

			case 'L':
				pargs-
>log_path = ptr+2;
				break;

			case 'p':
				pargs-
>pack_size = atol(ptr+2);
				break;

			case 'i':
				pargs-
>build_index = atol(ptr+2);
				break;

			case 'o':

				pargs-
>index_order = atol(ptr+2);
				break;

			case 'c':
				pargs-
>scale_down = atol(ptr+2);
				break;

		case 'd':
				pargs-
>index_script_path = ptr+2;
				break;

			default:

	GetArgsLoaderUsage();
			exit(-1);
			break;
		}
    }

    /* check for required args */
    if (pargs->num_warehouses == UNDEF )
    {
		printf("Number of
Warehouses is required\n");
		exit(-2);
    }

    return;
}


//=========================================
================================
//
// Function name: GetArgsLoaderUsage
//
//=========================================
================================

void GetArgsLoaderUsage()
{

#ifdef DEBUG
    printf("[%ld]DBG: Entering
GetArgsLoaderUsage()\n", (int)
GetCurrentThreadId());
#endif


    printf("TPCCLDR:\n\n");
	printf("Parameter
Default\n");
	printf("-------------------------
------------------------------------
\n");
    printf("-W Number of Warehouses to
Load                       Required \n");
    printf("-S Server
%s\n", SERVER);
    printf("-U Username
%s\n", USER);
    printf("-P Password
%s\n", PASSWORD);
    printf("-D Database
%s\n", DATABASE);
	printf("-b Batch Size
%ld\n", (long) BATCH);
	printf("-p TDS packet size
%ld\n", (long) DEFLDPACKSIZE);
	printf("-f Loader Results Output
Filename                    %s\n",
LOADER_RES_FILE);
	printf("-s Starting Warehouse
%ld\n", (long) DEF_STARTING_WAREHOUSE);
	printf("-i Build Option (data =
0, data and index = 1)       %ld\n",
(long) BUILD_INDEX);
	printf("-o Cluster Index Build
Order (before = 1, after = 0)  %ld\n",
(long) INDEX_ORDER);
	printf("-c Build Scaled Database
(normal = 0, tiny = 1)       %ld\n",
(long) SCALE_DOWN);
	printf("-d Index Script Path
%s\n", INDEX_SCRIPT_PATH);
	printf("-t Table to Load
all tables \n");
    printf("
[item|warehouse|customer|orders]\n");
    printf("   Notes: \n");
    printf("     - the '-t' parameter may
be included multiple times to \n");
    printf("       specify multiple
tables to be loaded \n");
    printf("     - 'item' loads ITEM
table \n");
    printf("     - 'warehouse' loads
WAREHOUSE, DISTRICT, and STOCK tables
\n");
    printf("     - 'customer' loads
CUSTOMER and HISTORY tables \n");
    printf("     - 'orders' load NEW-
ORDER, ORDERS, ORDER-LINE tables \n");
```

```
        printf("\nNote:  Command line
switches are case sensitive.\n");

    exit(0);
}
```

Appendix C:
Tunable Parameters

Server Configuration Parameters

Microsoft Windows .NET Enterprise Server Parameters

The following registry key was added:
[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\I/O System]
"CountOperations"=dword:00000000

Microsoft Windows .NET Enterprise Server Configuration

The following services were disabled on the server:

- Alerter
- Automatic Updates
- Computer Browser
- Cryptographic Services
- DHCP Client
- Distributed File System
- Distributed Link Tracking Client
- DNS Client
- Global Array Manager Server
- Help and Support
- IPSEC Policy Agent
- License Logging Service
- Messenger
- MSSQLserver
- Microsoft Search
- Print Spooler
- Process Control Service
- Remote Registry Service
- Removable Storage
- Run as Service
- System Event Notification
- SSDP Discovery service
- Task Scheduler
- Wireless configuration

Microsoft SQL Server 2000 Startup Parameters

Microsoft SQL Server was started with the following command line options

sqlservr -c -x -T3502 –g100

where

-c	Start SQL Server independently of the Microsoft Windows NT Service Control Manager.
-x	Disable the keeping of CPU time and cache-hit ratio statistics.
-T3502	Prints a message to the log at the beginning and end of each checkpoint.
-g100	Reserve 100 MB for non-buffer pool allocations

Boot.ini Parameters

The flag /pae was added to the boot line in the boot.ini file.

Microsoft SQL Server 2000 Configuration Parameters

affinity mask	15
allow updates	1
awe enabled	1
lightweight pooling	1
locks	0
max degree of parallelism	1
max worker threads	385
min memory per query	1024
priority boost	1
recovery interval	60
remote access	0
remote login timeout	20
remote query timeout	600
set working set size	0

all other SQL Server configuration settings are default.