

# **amdahl**

---

---

## **TPC Benchmark™ C Full Disclosure Report**

**Amdahl**

  
**ENVISTA™ Frontline Server**

**using**

**Microsoft® SQL Server 6.5**

**and**

**Microsoft Windows® NT 4.0**

First Edition

Submitted for Review  
March 28, 1997

## **First Printing, March 28, 1997**

Amdahl Corporation 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. Furthermore, Amdahl Corporation is not responsible for any errors contained within this document.

The pricing information given in this FDR is accurate as of the publication date, March 28, 1997, but Amdahl Corporation cannot guarantee that all sources will offer the same pricing.

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

Copyright 1997 Amdahl Corporation

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

Microsoft, Windows NT and SQL Server for Windows NT are registered trademarks of Microsoft Corporation.

EnVista and Frontline Server are trademarks of Amdahl Corporation.

Intel and Pentium Pro Processor are registered trademarks of Intel Corporation.

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

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

# Abstract

---

## Overview

This report documents the methodology and results of the TPC Benchmark™ C test conducted on the Amdahl Corporation EnVista Frontline Server (FS). The tests were conducted by Amdahl Corporation.

The tests were run in a client/server configuration using six HiQ 133MHz Pentium® Personal Computers as clients. The operating system used for the benchmark was Microsoft NT Server 4.0 for both server and clients. The database was the Microsoft SQL Server v. 6.5.SP3.

All tests were done in compliance with Revision 3.2 of the Transaction Processing Council's TPC Benchmark™ C Standard Specification. Two standard TPC Benchmark™ C metrics, transactions per minute (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	Availability Date
Amdahl EnVista Frontline Server	Microsoft Windows NT 4.0 Microsoft SQL Server 6.5 SP3	\$593,988	7573.00	78.43	HW: March 1, 1997 SW: March 31, 1997

## AUDITOR

The results of the benchmark and test methodology used to produce the results were audited by Tom Sawyer of Performance Metrics and have fully met the TPC-C rev 3.2 specifications.

Additional copies of this Full Disclosure Report can be obtained from either the Transaction Processing Performance Council or Amdahl Corporation at the following address:

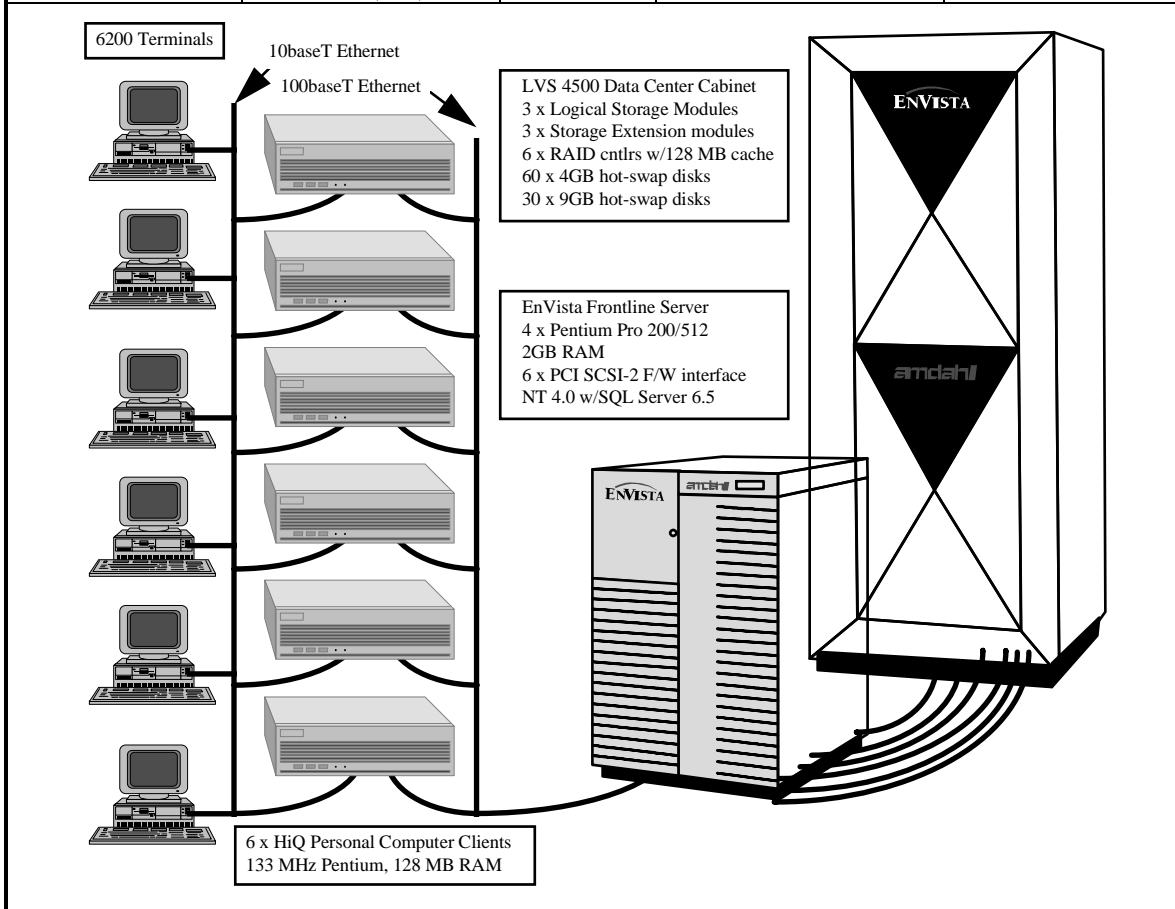
Transaction Processing Performance Council (TPC)  
c/o Shanley Public Relations  
777 North First Street, Suite 600  
San Jose, CA 95112, USA  
Phone: (408) 295-8894, fax 295-9768

or

Amdahl Corporation  
1250 East Arques Avenue  
PO Box 3470  
Sunnyvale, CA 94088-3470

Attn: John Howell M/S 358

Amdahl Corporation		EnVista Frontline Server™		TPC-C Rev 3.2 Report Date March 25, 1997	
Total System Cost		TPC-C Throughput	Price/Performance		Availability Date
<b>\$593,988</b>		<b>7573 tpmC</b>	<b>\$78.43 /tpmC</b>		<b>March 31, 1997</b>
Processors	Database Manager	Operating System	Other Software		Number of Users
4 200-MHz Pentium® Pro Processors	Microsoft SQL Server 6.5 SP 3 (246)	Microsoft Windows NT 4.0	Microsoft Internet Information Server Microsoft Visual C++		6200



System Component	Server		Clients	
Processors	4	Pentium® Pro Processor 200MHz	1	Pentium® 133MHz
Cache		512KB		256KB
Memory		2048MB		128MB
Disk Controllers	7	4 SCSI-2 Differential 1 SCSI-2 Diff (twintail) 2 Adaptec 7880	1	Adaptec 2940UW PCI
Disk Drives	36	9 GB	1	2 GB
	61	4 GB		
Total Storage	97	568 GB	1	2 GB
Other	1	CD-ROM	1	CD-ROM

Amdahl Corporation		EnVista Frontline Server			TPC-C REV 3.2 EXECUTIVE SUMMARY PAGE 2 OF 2		
		Client/Server			Report Date: 28-February-1997		
Description	Part Number	Third Party	Unit Price	Qty	Extended Price	5 yr. Maint. Price	
<b>Server Hardware</b>							
EnVista Base Cabinet	NVISTA-FS-4001		\$6,927	1	6,927	5,160	
200MHz/512KB Envista	NVISTA-FS-2654		\$8,644	1	8,644	1,920	
2 GB Memory	NVISTA-FS-2661		\$32,004	1	32,004	16,020	
PRO 100B PCI NIC	NVISTA-FS-2683		\$84	1	84	300	
SCSI-2 Differential Adapter	NVISTA-FS-2689		\$639	4	2,558	1,200	
SCSI-2 Differential Adapter (twin-tailed)		Adaptec	\$998	1	998	300	
Monitor 15"	NVISTA-MON-2720		\$441	1	441	600	
SVGA Adapter	NVISTA-SVGA-2721		\$227	1	227	240	
Keyboard/Mouse	NVISTA-KBD-2722		\$81	1	81	0	
CD-ROM	NVISTA-FS-2668		\$265	1	265	240	
Tape Backup	NVISTA-FS-2704		\$1,459	1	1,459	480	
4 GB hot-swappable disks	NVISTA-FS-2659		\$1,298	1	1,298	480	
9 GB hot-swappable disks	NVISTA-FS-2660		\$2,158	6	12,948	7,200	
<b>SUB-TOTAL</b>					<b>67,934</b>	<b>34,140</b>	
<b>Server Software</b>							
NT 4.0 Serverw/10 CAL	NVISTA-WN4020-0109		\$854	1	854	0	
SQL Server 6.5 w/5 CAL (*)	NVISTA-SQL655-0299		\$1,399	1	1,399	10,475	
Internet Connector License		Microsoft	\$2,999	1	2,999	0	
<b>SUB-TOTAL</b>					<b>5,252</b>	<b>10,475</b>	
<b>Storage</b>							
LVS 4500 Logical Storage Module	4500-001-8120		\$0	3	0	57,600	
288 MB Intelligent Memory	4500-001-7383		\$13,539	3	40,617	0	
Data Center Cabinet	4500-001-7371		\$3,115	1	3,115	0	
Storage Expansion Module	4500-001-7410		\$2,485	3	7,456	14,400	
SCSI Host Interface Cable (10 ft)	4500-001-8432		\$85	6	510	0	
UPS	EXIDE-UPS2D-6450		\$2,651	1	2,651	2,400	
9 GB hot-swappable Disk Drives (20 count)	4500-001-7989		\$77,377	1	77,377	0	
9 GB hot-swap Extension Disks	4500-001-7944		\$2,958	10	29,579	0	
4 GB hot-swappable Disk Drives (20 count)	4500-001-7405		\$49,090	2	98,179	0	
4 GB hot-swap Extension Disks	4500-001-7444		\$1,544	20	30,870	0	
<b>SUB-TOTAL</b>					<b>290,354</b>	<b>74,400</b>	
<b>Client Hardware</b>							
1P 133 MHz Pentium **			\$3,999	8	31,992		
2 GB disk							
128 MB memory							
PRO 100B 10/100 baseT NIC (2)							
8X CD-ROM							
SVGA Adapter							
Keyboard/mouse/15" monitor							
<b>SUB-TOTAL</b>					<b>31,992</b>		
<b>Client Software</b>							
NT 4.0 Server w/10 CAL		Microsoft	\$854	6	5,124		
SQL Server Pgmr Toolkit		Microsoft	\$499	1	499		
Visual C++ V 4.0		Microsoft	\$499	1	499		
<b>SUB-TOTAL</b>					<b>6,122</b>	<b>0</b>	
<b>User Connectivity</b>							
NETLUX 100baseT Hub **	8-port hub	NETLUX	\$595	3	1,785		
NETLUX 10baseT 24-port Hub **	24-port hub	NETLUX	\$251	285	71,535		
<b>SUB-TOTAL</b>					<b>73,320</b>	<b>0</b>	
<b>TOTAL</b>					<b>\$474,973</b>	<b>\$119,015</b>	
(*) Maintenance of Microsoft software products is covered by the SQL Server maintenance					<b>Five-Year Cost of Ownership: \$593,988</b>		
(**) 5-year warranty with 10%-spares							
Price sources: 1=Amdahl 2=Microsoft 3=HIQ Computers 4=Netlux					<b>tpmC Rating: 7573.00</b>		
<b>Audited by Tom Sawyer of Performance Metrics, Inc</b>					<b>\$ / tpmC: 78.43</b>		
Prices used in TPC benchmarks reflect the actual prices a customer would pay for a one-time purchase of the stated components. Individually negotiated discounts are not permitted. Special prices based on assumptions about past or future purchases are not permitted. All discounts reflect standard pricing policies for the listed components. For complete details, see the pricing sections of the TPC benchmark specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org.							

**MQTh**, computed Maximum Qualified Throughput 7573 tpmC  
 % throughput difference, reported & reproducibility runs 0.4%

**Response Times** (90th percentile | Average | Maximum) in seconds

- Neworder	1.2	0.7	6.6
- Payment	1.1	0.6	3.5
- Order Status	1.8	1.3	7.2
- Delivery (interactive portion)	0.3	0.3	0.8
- Delivery (deferred portion)	1.4	1.2	14.4
- Stock-Level	3.6	2.4	8.5
- Menu	0.3	0.2	0.9

Response time delay added for menu delay 0.1

Response time delay added for response time delay 0.1

**Transaction Mix**, in percent of total transactions

- New-Order	44.7 %
- Payment	43.1 %
- Order-Status	4.1 %
- Delivery	4.1 %
- Stock-Level	4.1 %

**Keying/Think Times** (in seconds),

	Min		Average		Max	
- New-Order	18.0	0	18.0	12.1	18.1	120.8
- Payment	3.0	0	3.0	12.1	3.0	120.7
- Order-Status	2.0	0	2.0	10.2	2.0	100.7
- Delivery	2.0	0	2.0	5.1	2.0	50.7
- Stock-Level	2.0	0	2.0	5.1	2.0	44.8

**Test Duration**

- Ramp-up time	25 minutes
- Measurement interval	30 minutes
- Number of checkpoints	1
- Checkpoint interval	30 minutes
- Number of transactions (all types) completed in measurement interval	508,268

(and all other numerical quantities required in the Full Disclosure Report)

# Introduction

---

## **Document Structure**

The contents of this report are determined by the TPC Benchmark C Standard Specification Revision 3.2, written and approved by the Transaction Processing Performance Council (TPC). The format of this report is based on this specification. Most sections of this report begins with the relevant specification requirements printed in *italic type*, immediately followed by the detail in plain type of how Amdahl Corporation complied with the specification. Where extensive listings are required (such as listing of code), a note is included which references an appendix containing the listing.

## **Benchmark Overview**

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.

## **System Overview**

The hardware configuration used in this TPC-C test is the Amdahl EnVista Frontline Server (FS). The test configuration was built with 2GB of memory, various Adaptec controllers, Amdahl Large Volume Storage (LVS) 4500 RAID controllers, a combination of 4GB and 9GB disk drives, and Intel Pro100B Smart Network adapter cards. The operating system used was Microsoft NT 4.0 Server, and Microsoft SQL Server 6.5 SP3 was used as the database engine.

The EnVista FS motherboard is based on the Intel 82450GX chipset and contains two processor boards that can hold up to 4 Pentium Pro Processors (200MHz with 512K L2 cache). The system has 10 I/O slots based on dual-peer PCI bus. Six of the slots are PCI and four are EISA, and none are shared. The EnVista benchmark configuration used 2GB of RAM, consisting of 16 128MB SIMMs on the memory board. The EnVista FS incorporates I2O ready BIOS. This means that as intelligent I/O devices become available, the system can take advantage of their performance benefits. This system has two integrated Ultra/Wide SCSI-3 controllers (based on Adaptec AIC-7880) and offers a peak transfer rate of 40 MB/s each. In a standard configuration, one Ultra/Wide SCSI-3 controller is cabled to the hard drive backplane, and the other is cabled to the CD-ROM and removable media bays.

The Amdahl LVS 4500 RAID disk system was configured in a Data Center cabinet and loaded with 90 hot-pluggable disks. The LVS 4500 storage sub-system utilized three Logical Storage Modules (LSM), each configured with 288 MB of Intelligent Memory for the RAID controllers and cache. Each LSM was configured with two active controllers in non-mirrored mode, with 128 MB of cache (with integral battery backup) per controller. The Amdahl LVS 4500 controllers can host Ultra Wide SCSI-3 connections, although the benchmark environment described here utilized SCSI-2 Fast/Wide differential connections. Each LSM was configured with one Storage Expansion cabinet, and the total allocation of disks per dual-controller LSM was 30 disks.

The EnVista system also used the Intel Pro100B PCI network adapter card. This network interface card (NIC) supplied a 100Base T network interface to the six HiQ Personal Computer clients. Each of the clients had one 133MHz Pentium® processor, 128MB of Memory, one 2GB SCSI hard disk and two Intel Pro100B network adapter cards. One of the cards interfaced with the EnVista server at 100baseT speeds, while the other card interfaced to the users (e.g., the RTE systems) at 10baseT speeds.

All components of the benchmark configuration were configured in a manner appropriate to customer production environments, i.e., the system used full-function drivers and software. The exception is that operating system components unused during the benchmark were disabled (see Appendix C).



<b>ABSTRACT.....</b>	<b>3</b>
OVERVIEW.....	3
AUDITOR.....	3
<b>INTRODUCTION.....</b>	<b>7</b>
DOCUMENT STRUCTURE.....	7
BENCHMARK OVERVIEW.....	7
SYSTEM OVERVIEW.....	8
<b>GENERAL ITEMS.....</b>	<b>12</b>
TEST SPONSOR.....	12
APPLICATION CODE AND DEFINITION STATEMENTS.....	12
PARAMETER SETTINGS.....	12
CONFIGURATION DIAGRAMS.....	12
<b>CLAUSE 1 -- LOGICAL DATABASE DESIGN RELATED ITEMS .....</b>	<b>15</b>
TABLE DEFINITIONS.....	15
PHYSICAL ORGANIZATION OF THE DATABASE.....	15
INSERT AND DELETE OPERATIONS.....	15
HORIZONTAL AND VERTICAL PARTITIONING.....	15
REPLICATION.....	15
TABLE ATTRIBUTES.....	15
<b>CLAUSE 2 -- TRANSACTION AND TERMINAL PROFILES RELATED ITEMS .....</b>	<b>16</b>
RANDOM NUMBER GENERATION.....	16
SCREEN LAYOUT.....	16
TERMINAL VERIFICATION.....	16
INTELLIGENT TERMINALS.....	16
TRANSACTION PROFILES.....	17
TRANSACTION MIX.....	17
DEFERRED DELIVERY MECHANISM.....	17
<b>CLAUSE 3 -- TRANSACTION AND SYSTEM PROPERTIES RELATED ITEMS .....</b>	<b>18</b>
ACID TESTS.....	18
<i>Atomicity</i> .....	18
<i>Consistency</i> .....	18
<i>Isolation</i> .....	18
<i>Durability</i> .....	19
<b>CLAUSE 4 -- SCALING AND DATABASE POPULATION RELATED ITEMS .....</b>	<b>20</b>
TABLE CARDINALITY.....	20
CONSTANT VALUES.....	20
DATA DISTRIBUTION.....	21
PARTITION MAPPING.....	22
180 DAY SPACE CALCULATION.....	22
<b>CLAUSE 5 -- PERFORMANCE METRICS AND RESPONSE TIME RELATED ITEMS .....</b>	<b>23</b>
MEASURED TPMC.....	23
RESPONSE TIMES.....	23
THINK TIMES & KEY TIMES.....	23
RESPONSE TIME DISTRIBUTION CURVES.....	24

NEW-ORDER RESPONSE TIME VS. THROUGHPUT GRAPH.....	26
NEW-ORDER THINK TIME DISTRIBUTION GRAPH .....	27
STEADY-STATE GRAPH.....	28
STEADY-STATE METHODOLOGY.....	28
WORK PERFORMED DURING STEADY STATE .....	28
REPRODUCIBILITY METHODOLOGY.....	29
MEASUREMENT INTERVAL.....	29
TRANSACTION MIX.....	30
CHECKPOINTS.....	30
<b>CLAUSE 6 -- SUT, DRIVER, AND COMMUNICATION DEFINITION RELATED ITEMS .....</b>	<b>31</b>
RTE PARAMETERS .....	31
EMULATED COMPONENTS.....	31
BENCHMARKED AND TARGETED SYSTEM CONFIGURATION DIAGRAMS.....	31
NETWORK CONFIGURATION.....	31
NETWORK BANDWIDTH.....	31
OPERATOR INTERVENTION.....	32
<b>CLAUSE 7 -- PRICING RELATED ITEMS.....</b>	<b>33</b>
HARDWARE AND SOFTWARE LIST.....	33
AVAILABILITY DATE.....	33
MEASURED TPMC.....	33
COUNTRY SPECIFIC PRICING.....	33
USAGE PRICING .....	33
SYSTEM PRICING .....	33
<b>CLAUSE 9 -- AUDIT RELATED ITEMS.....</b>	<b>34</b>
AUDITOR.....	34
AVAILABILITY OF THE FULL DISCLOSURE REPORT.....	34
AUDITOR ATTESTATION LETTER.....	35
<b>APPENDIX A - APPLICATION SOURCE CODE .....</b>	<b>37</b>
MICROSOFT WEB CLIENT .....	37
<i>Makefile</i> .....	37
<i>Delisrv.h</i> .....	38
<i>Httpext.h</i> .....	39
<i>Resource.h</i> .....	40
<i>Tpcc.h</i> .....	40
<i>Trans.h</i> .....	43
<i>Delisrv.c</i> .....	46
<i>Tpcc.c</i> .....	57
<i>Webclnt.c</i> .....	100
<b>APPENDIX B - DATABASE DESIGN .....</b>	<b>105</b>
BUILD.....	105
<i>Createdb.sql</i> .....	105
<i>Diskinit.sql</i> .....	105
<i>Segment.sql</i> .....	105
<i>Tables.sql</i> .....	106
<i>Idxcuscl.sql</i> .....	107
<i>Idxcusnc.sql</i> .....	107
<i>Idxdiscl.sql</i> .....	107
<i>Idxitmcl.sql</i> .....	107
<i>Idxnodcl.sql</i> .....	107
<i>Idxodlcl.sql</i> .....	108

<i>Idxordcl.sql</i> .....	108
<i>Idxstkcl.sql</i> .....	108
<i>Idxwarcl.sql</i> .....	108
<i>Dbopt1.sql</i> .....	108
<i>Dbopt2.sql</i> .....	108
<i>Pintable.sql</i> .....	108
<i>Tpccbcpsql</i> .....	108
<i>Tpccirl.sql</i> .....	109
STORED PROCEDURES .....	109
<i>Neword.sql</i> .....	109
<i>Payment.sql</i> .....	110
<i>Delivery.sql</i> .....	112
<i>Ordstat.sql</i> .....	112
<i>Stocklev.sql</i> .....	113
<b>APPENDIX C - TUNING PARAMETERS</b> .....	<b>114</b>
MICROSOFT WINDOWS NT SERVER TUNABLE PARAMETERS .....	114
<i>Registry Changes</i> .....	114
<i>Microsoft SQL Server version 6.5 Startup Parameters</i> .....	114
<i>DBCC GAMINIT</i> .....	114
<i>Microsoft Windows NT Server Configuration Parameters</i> .....	114
<i>Microsoft SQL Server 6.5 Configuration Parameters</i> .....	115
<i>Microsoft SQL Server Stack Size</i> .....	116
<i>Disk Array Configuration Parameters</i> .....	116
CLIENT CONFIGURATION PARAMETERS .....	116
<i>Microsoft Windows NT Server Configuration Parameters</i> .....	116
<i>RTE Parameters</i> .....	117
<i>RTE Pacing Sets</i> .....	120
<b>APPENDIX D - DISK STORAGE</b> .....	<b>122</b>
180 DAY SPACE CALCULATIONS .....	122
<b>APPENDIX E - PRICE QUOTES</b> .....	<b>123</b>

## General Items

---

### **Test Sponsor**

*A statement identifying the sponsor of the Benchmark and any other companies who have participated.*

Amdahl Corporation is the sponsor of this TPC Benchmark™ C.

### **Application Code and Definition Statements**

*The application program must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input/output functions.*

Section 3.0, entitled Clause 3 Related Items, contains a brief discussion of the database design and loading procedures. The database definition statements, distribution across disk drives, loading scripts, and tables are provided in Appendix A-Database Generation.

The program that implements the TPC Benchmark C translation and collects appropriate transaction statistics is referred to as the Remote Terminal Emulator (RTE) or Driver program. The Driver program is discussed in Section 7.0. The source code for this driver program is provided in Appendix B - Source code.

### **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 options*
- *Recovery/commit options*
- *Consistency/locking options*
- *System parameters, application parameters, and configuration parameters.*

*This requirement can be satisfied by providing a full listing of all parameters and options.*

Appendix C contains the database and operating system parameters used in this benchmark. Appendix D contains the hardware configuration details.

### **Configuration Diagrams**

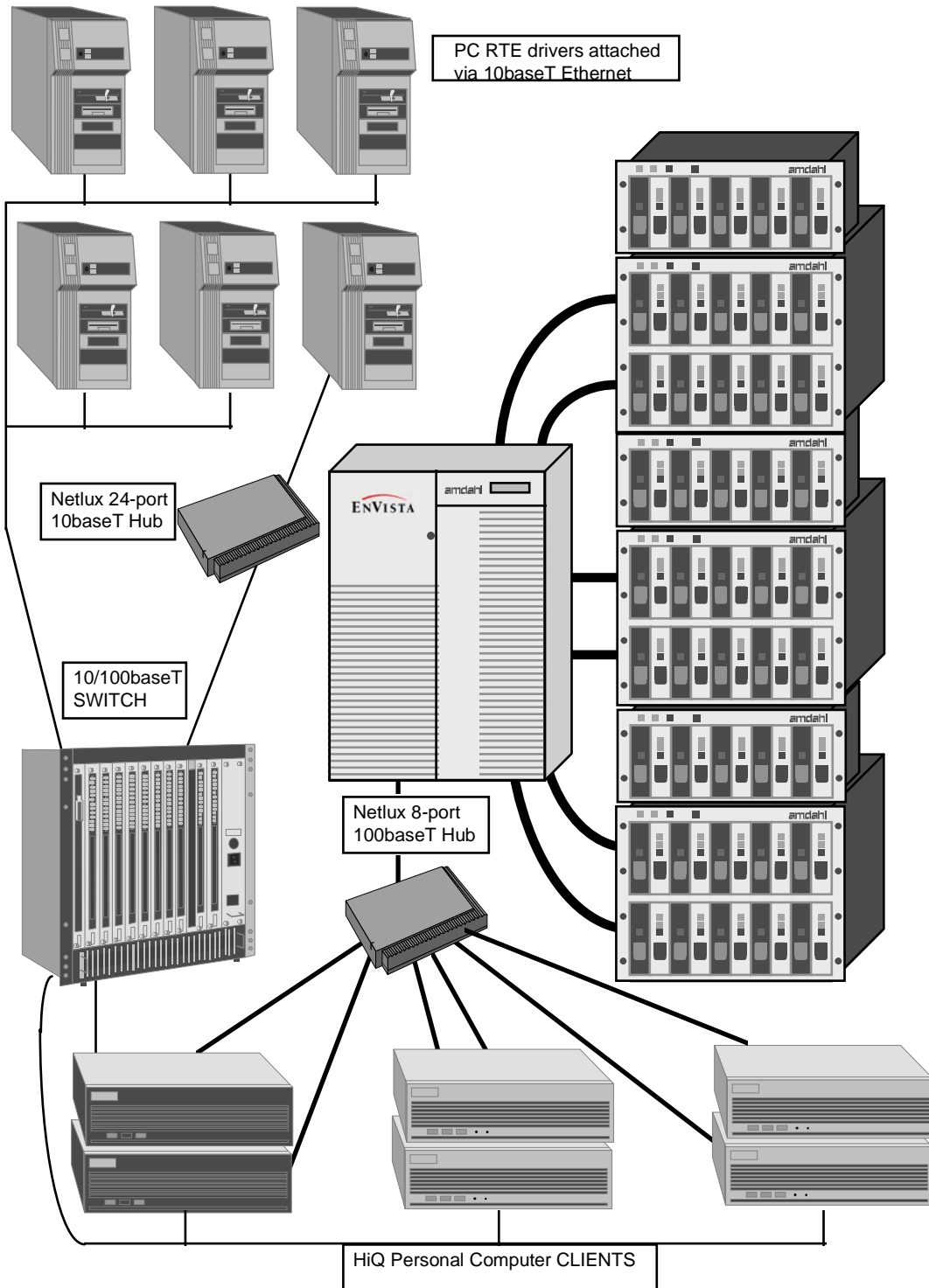
*Diagrams of both the measured and priced systems must be provided, accompanied by a description of the differences.*

Figure 1-1 and 1-2 respectively show the measured and priced full client/server configurations. The differences between the measured and priced configurations are highlighted below.

There were 15 9GB drives for the data and 4 for the log on the measured configuration, and 30 for data and 6 for the log on the priced configuration.

The measurement configuration utilized a 16-port 10/100baseT switch to connect the RTE machines to the clients. The switch was configured as 10baseT, half-duplex. There was no gain in performance while using the switch, demonstrated by inserting a single 10baseT Ethernet hub into one RTE-to-Client connection: The individually recorded response times were statistically identical between switch and hub connections.

The Client machines were benchmarked with 4GB disk drives, but 2GB drives were priced. There is no performance advantage inherent in Client disk capabilities.



**Figure 1: Measured Configuration**



## Clause 1 -- Logical Database Design Related Items

---

### **Table Definitions**

*Listings must be provided for all table definition statements and all other statements used to set-up the database (from Clause 8.1.2.1).*

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 (Clause 8.1.2.2)*

The measured configuration used ninety (96) SCSI-2 Fast/Wide disk drives. Ninety of the drives were installed in the Amdahl LVS 4500 Data Center Cabinet, of which there were fifteen 9GB drives and seventy-five 4GB drives. The remaining six drives were installed in the EnVista Frontline server and were used as LOG files (6) and the operating system filesystem.

The ninety drives in the LVS 4500 were configured as six groups of 15 disks each, all of them RAID 0 device groups. Each group of 15 drives was connected to the EnVista host via a dedicated SCSI-2 Fast/Wide differential connection.

### **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 cardinality 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 (Clause 8.1.2.3)*

Insert and delete functionality was fully operational during the benchmark.

### **Horizontal and 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 (Clause 8.1.2.4).*

Partitioning was not used in this benchmark.

### **Replication**

*Replication of tables, if used, must be disclosed (see Clause 1.4.6) (from Clause 8.1.2.5)*

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) (Clause 8.1.2.6).*

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 described (Clause 8.1.3.1)*

The random number generation was performed internal to the Microsoft BenchCraft RTE program, which was audited independently.

### **Screen Layout**

*The actual layouts of the terminal input/output screens must be disclosed (Clause 8.1.3.2).*

The screen layouts are based on those in Clauses 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3 of the TPC-C Standard Specification. There are some differences due to the fact this is a WEB client implementation.

### **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). (Clause 8.1.3.3)*

The terminal features were verified by allowing the auditor to manually execute each of the five transaction types using Microsoft Internet Explorer version 3.0.

### **Intelligent Terminals**

*Any usage of presentation managers or intelligent terminals must be explained. (Clause 8.1.3.4).*

**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.*

Application code involved in the manipulation of data was run on the client. Screen manipulation commands were downloaded to the WEB browser, which handled input and output presentation graphics. A listing of this code is included in Appendix A. Microsoft Internet Information Service® assisted in the processing and presentation of this data.



## Transaction Profiles

The percentage of home and remote order-lines in the New-Order transactions must be disclosed (Clause 8.1.3.5). The percentage of New-Order transactions that rolled-back as a result of an unused item number must be disclosed (Clause 8.1.3.6). The number of items per orders entered by New-Order transactions must be disclosed (Clause 8.1.3.7). The percentage of home and remote Payment transactions must be disclosed (Clause 8.1.3.8). The percentage of Payment and Order-Status transactions that used non-primary key (C\_LAST) access to the database must be disclosed (Clause 8.1.3.9). 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 1: Transaction Statistics**

Transaction	Function	Value
New Order	Home Warehouse Items	98.99%
	Remote Warehouse Items	1.01%
	Rolled Back Transactions	1.01%
	Average Lines Per Order	10
Payment	Home Warehouse	85.02%
	Remote Warehouse	14.98%
	Non-Primary Key Access	60.07%
Order Status	Non-Primary Key Access	60.09%
Delivery	Skipped Transactions	0

## Transaction Mix

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

**Table 2: Transaction Mix**

Transaction	Percentage
New Order	44.70%
Payment	43.10%
Order Status	4.07%
Delivery	4.05%
Stock Level	4.08%

## Deferred Delivery Mechanism

The queuing mechanism used to defer the execution of the Delivery transaction must be disclosed (Clause 8.1.3.12).

The client application processes submitted delivery transactions to named pipe delivery server software running on the client machines. There was a single delivery server with multiple execution threads running on each client machine. These delivery servers were responsible for processing deliveries queued to the named pipe and submitting them to the database server.

The source code is listed in Appendix A.

## Clause 3 -- Transaction and System Properties Related Items

---

### **ACID Tests**

*The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosing which case was followed for the execution of Isolation Test 7 (Clause 8.1.4.1).*

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

### **Atomicity**

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

### **Completed Transactions**

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

### **Aborted Transactions**

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

### **Consistency**

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

Consistency conditions One through Four were tested using a shell 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 ten (10) minutes and included a checkpoint. The shell script was executed again. The result of the same queries verified that the database remained consistent after the run.

### **Isolation**

*Sufficient conditions must be enabled at either the system or application level to ensure the required isolation defined in Clause 3.4.1 is obtained.*

Isolation tests One through Seven 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 verified the captured files. This demonstrated the required isolation had been met.

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

For Isolation test seven, case A was followed.

## **Durability**

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

### **Durable Media Failure**

Durability from media failure was demonstrated on a 10 warehouse database for the loss of Data and Log and the fully scaled database for the Instantaneous Interruption and Loss of Memory. The standard driving mechanism was used to generate the transaction load of 100 users for the Loss of Data and Log and 6200 users for the Instantaneous Interruption and Loss of Memory.

### **Loss of Data and Log**

The loss of data and log tests were performed on a 10 warehouse system. To demonstrate recovery from a permanent failure of durable medium containing TPC-C tables, the following steps were executed:

1. A 10 warehouse database was built in a manner similar to the full size database.
2. A backup was taken of the database using Microsoft SQL Server facilities.
3. A sum of d\_next\_o\_id was taken and noted.
4. 100 users were logged into the database and proceeded to run transactions.
5. Removed one disk drive from a disk array containing the transaction log.
6. NT reported that part of a mirrored pair was not responding.
7. SQL Server reported no errors and continued running normally.
8. Continued to run for 2 min.
9. Removed one disk drive from a disk array containing database files.
10. SQL Server reported errors and transactions failed.
11. SQL Server was restarted and a dump of the transaction log was taken.
12. The tpcc database was dropped.
13. The system was shut down and the removed disk drive was replaced and the partition reformatted.
14. The 10 warehouse database was recreated.
15. The 10 warehouse database was restored from backup.
16. The saved transaction log was loaded and transactions rolled forward.
17. A sum of d\_next\_o\_id was taken and noted.
18. The number of transactions reported in the database and in the RTE log was compared.

### **Instantaneous Interruption and Loss of Memory**

Because loss of power erases the contents of memory, the instantaneous interruption and the loss of memory tests were combined into a single test. This test was executed on a fully scaled database of 620 warehouses under a full load of 6200 users. The following steps were executed:

1. A sum of D\_NEXT\_O\_ID was taken and noted.
2. 6200 users were logged into the database and proceeded to run transactions.
3. System power was terminated.
4. System power was reapplied and NT restarted.
5. SQL Server was started and recovery occurred automatically.
6. A new sum of D\_NEXT\_O\_ID was taken and noted.
7. The number of transactions reported in the database and in the RTE log was compared.

## Clause 4 -- Scaling and Database Population Related Items

---

### **Table Cardinality**

The cardinality (e.g., the number of rows) of each , 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 (Clause 8.1.5.1).

The database was originally built with 650 warehouses. Thirty (30) rows were removed from the benchmark WAREHOUSE table prior to the run per Clause 4.2.2 of the TPC-C version 3.2 specification.

**Table 3: Table Cardinality**

<b>Table</b>	<b>Cardinality as Benchmarked</b>
Warehouse	620
District	6,500
Customer	19,500,000
History	19,500,000
Orders	19,500,000
New Order	5,850,000
Order Line	195,004,080
Stock	65,000,000
Item	100,000
Deleted Warehouses	30

### **Constant Values**

The following values were used as constant value inputs to the NURand function for this benchmark.

**Table 4: Constant Values**

<b>Function</b>	<b>Constant C Value</b>
C_LAST (Build)	123
C_LAST (Run)	233

## Data Distribution

The distribution of tables and logs across all media must be explicitly depicted for the tested and priced systems (Clause 8.1.5.2).

**Table 5 : Data Distribution**

Controller	Drives	Partition	size	Use
LVS1	15 x 9GB	E:	14,000MB	Misc_seg
		F:	18,000MB	Ol_seg
LVS2	15 x 4GB	G:	8,000MB	Cs_seg
LVS3	15 x 4GB	H:	8,000MB	Cs_seg
LVS4	15 x 4GB	I:	8,000MB	Cs_seg
LVS5	15 x 4GB	J:	8,000MB	Cs_seg
LVS6	15 x 4GB	K:	8,000MB	Cs_seg
		Y:	127,000MB	Backup
Adaptec	4GB	C:	4,000MB	OS
	9GB		8,600MB	Log_seg
	9GB	(mirror)	8,600MB	Log_seg
	9GB		8,600MB	Log_seg
	9GB	(mirror)	8,600MB	Log_seg
	4GB		4,000MB	Log_seg
	4GB	(mirror)	4,000MB	Log_seg

Clause 8.1.5.3: A statement must be provided that describes:

1. The data model implemented by the DBMS (e.g., relational, network, hierarchical)
2. The database interface (e.g., embedded, call level) and access language (e.g., SQL, DL/I, 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 v6.50.246 is a relational DBMS.

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

## **Partition Mapping**

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

**Comment:** *The intent is to provide sufficient detail about partitioning and replication to allow independent reconstruction of the test database.*

The database was not replicated.

## **180 Day Space Calculation**

*Details of the 180 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) (Clause 8.1.5.5).*

We calculated the 180 day space in the following manner:

1. The free space on the log file was queried using DBCC checktable(syslogs).
2. Transactions were run against the database with a full load of users.
3. The free space was again queried using DBCC checktable(syslogs).
4. The space used was calculated as the difference between the first and second query.
5. The number of NEW-ORDERS was verified from an RTE report covering the entire run.
6. The space used was divided by the number of NEW-ORDERS giving a space-used per NEW-ORDER transaction.
7. The space used per transaction was multiplied by the measured tpmC rate times 480 minutes.

The results of the above steps yielded a requirement of 21.16 GB (including mirror) to sustain the log for 8 hours. Space available on the transaction log volume was 34.4 GB (including mirror), indicating that enough storage was configured to sustain 8 hours of growth.

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

The details of the 180-day space requirements are shown in Appendix D.

## Clause 5 -- Performance Metrics and Response Time Related Items

---

### Measured tpmC

Measured tpmC must be reported. (8.1.6.1)

Measured tpmC                7573.00  
 Price per tpmC                \$78.43

### Response Times

Ninetieth percentile, maximum, and average response times must be reported for all transactions types as well as for the Menu response time (Clause 8.1.6.2).

**Table 6: Transaction Response Times**

Transaction	Average	Maximum	90%
New Order	0.71	6.58	1.20
Payment	0.57	3.45	1.10
Order Status	1.26	7.19	1.80
Interactive Delivery	0.27	0.75	0.30
Deferred Delivery	1.19	14.35	1.37
Stock Level	2.41	8.49	3.60
Menu	019	0.88	0.32

### Think Times & Key Times

The minimum, average, and maximum keying and think times must be reported for each transaction type (Clause 8.1.6.3).

**Table 7: Transaction Key Times**

Transaction	Average	Minimum	Maximum
New Order	18.02	18.02	18.07
Payment	3.03	3.02	3.07
Order Status	2.03	2.02	2.07
Delivery	2.03	2.02	2.05
Stock Level	2.03	2.02	2.08

**Table 8: Transaction Think Times**

Transaction	Average	Minimum	Maximum
New Order	12.09	0	120.80
Payment	12.06	0	120.70
Order Status	5.11	0	50.70
Delivery	5.06	0	44.76
Stock Level	10.17	0	100.70

### Response Time Distribution Curves

Response Time frequency distribution curves (see Clause 5.6.1) must be reported for each transaction type. (Clause 8.1.6.4)

Figure 3: New Order Response Time Distribution

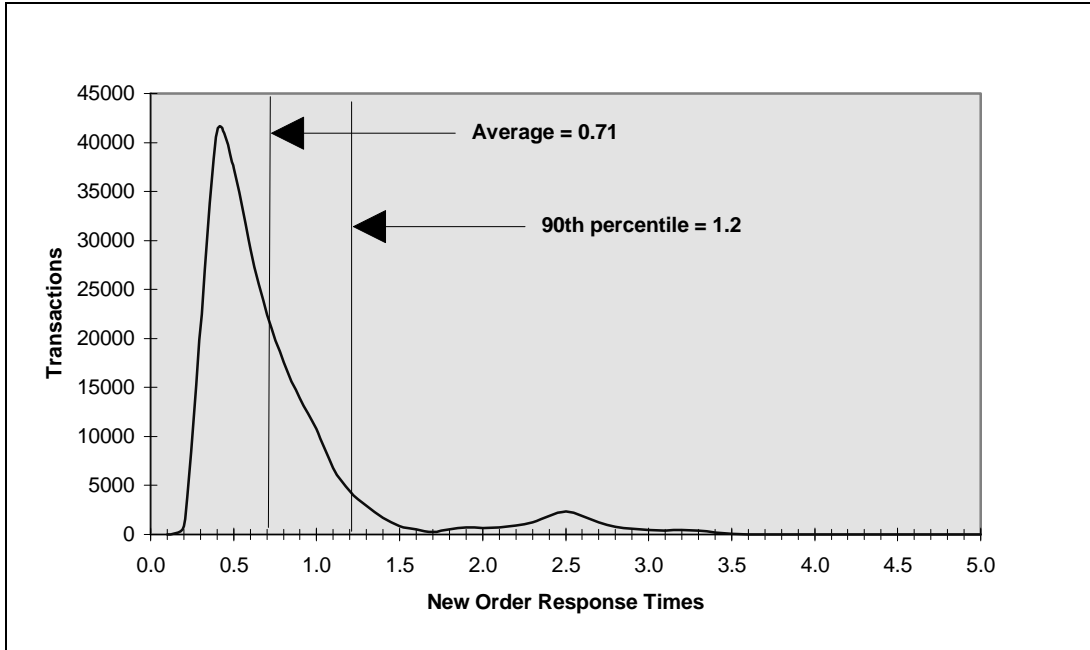
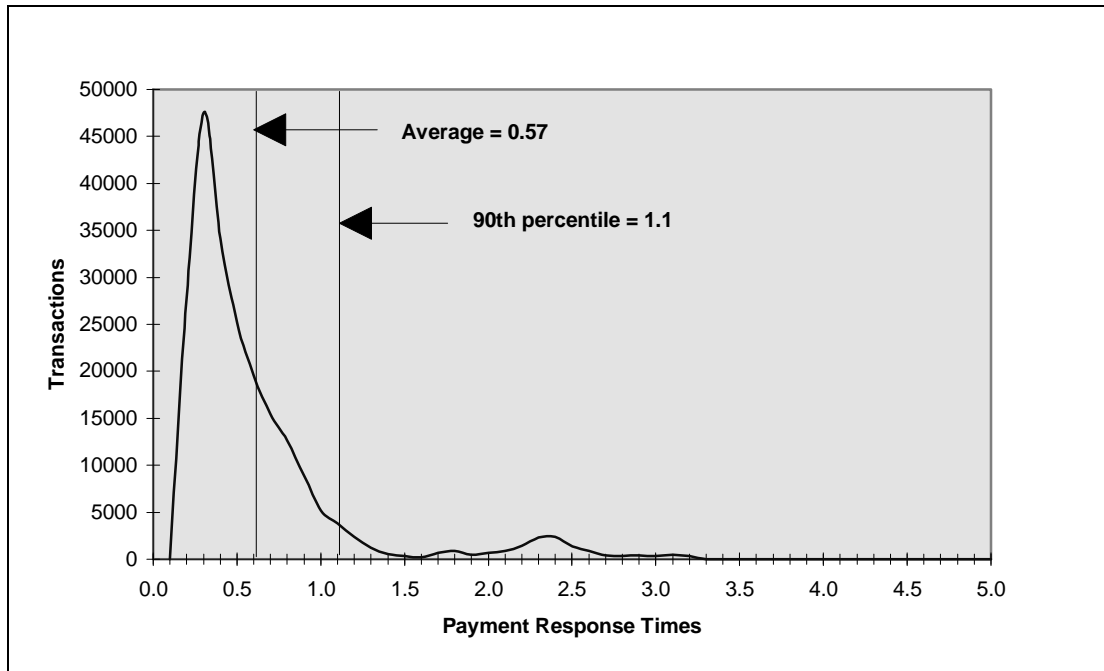
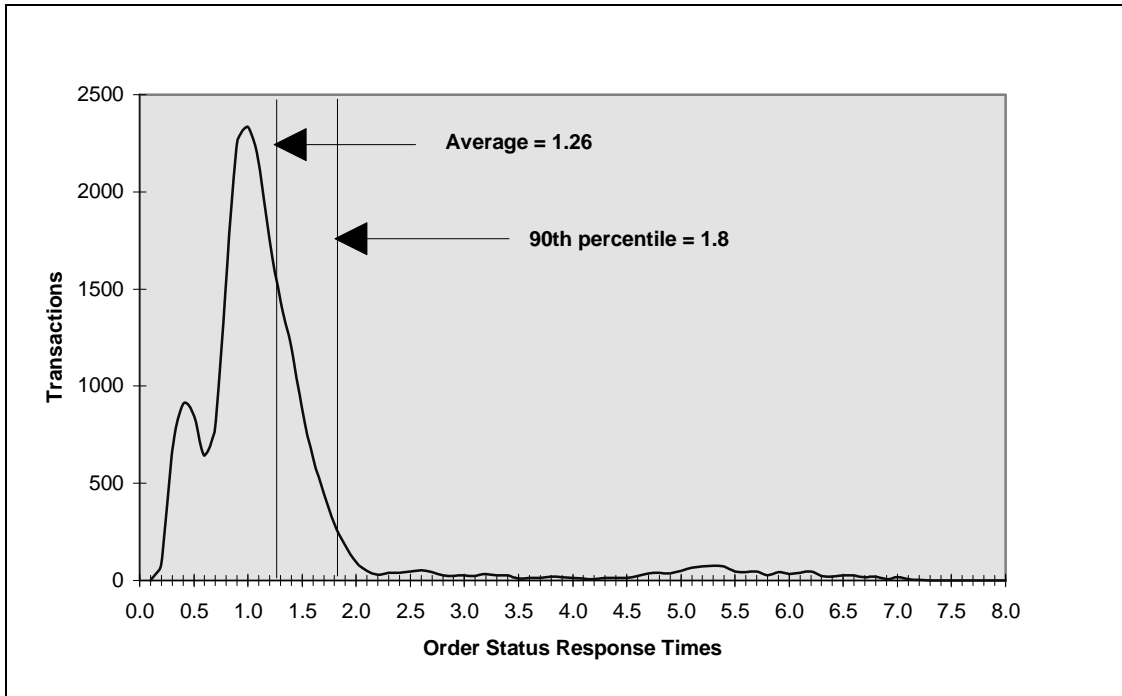


Figure 4: Payment Response Time Distribution

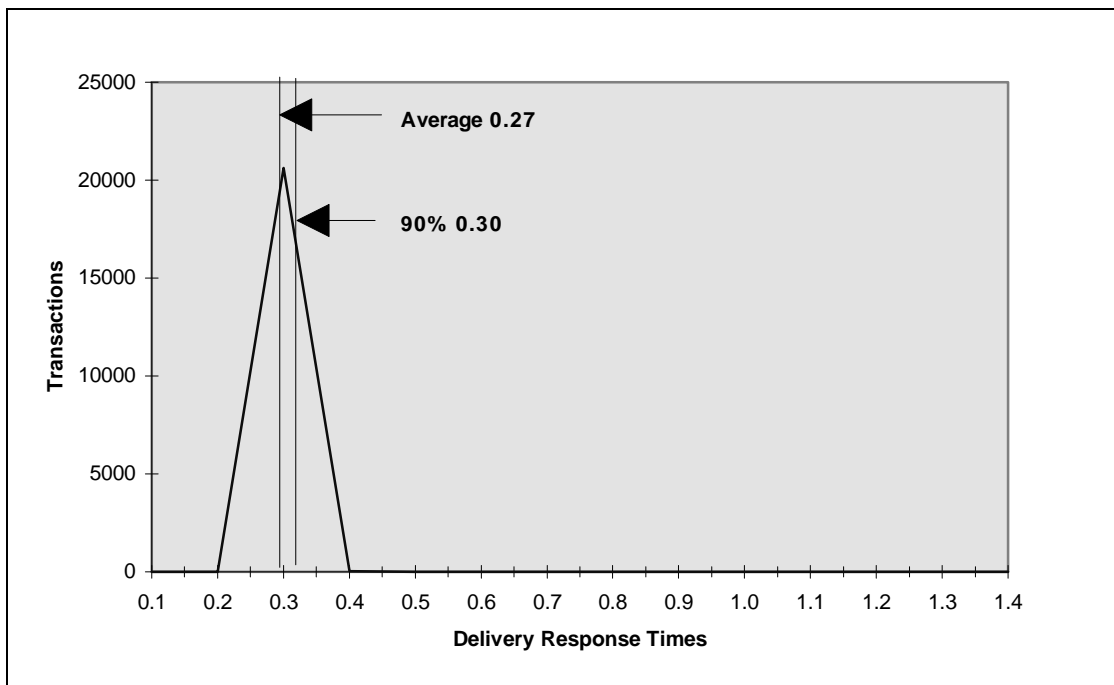




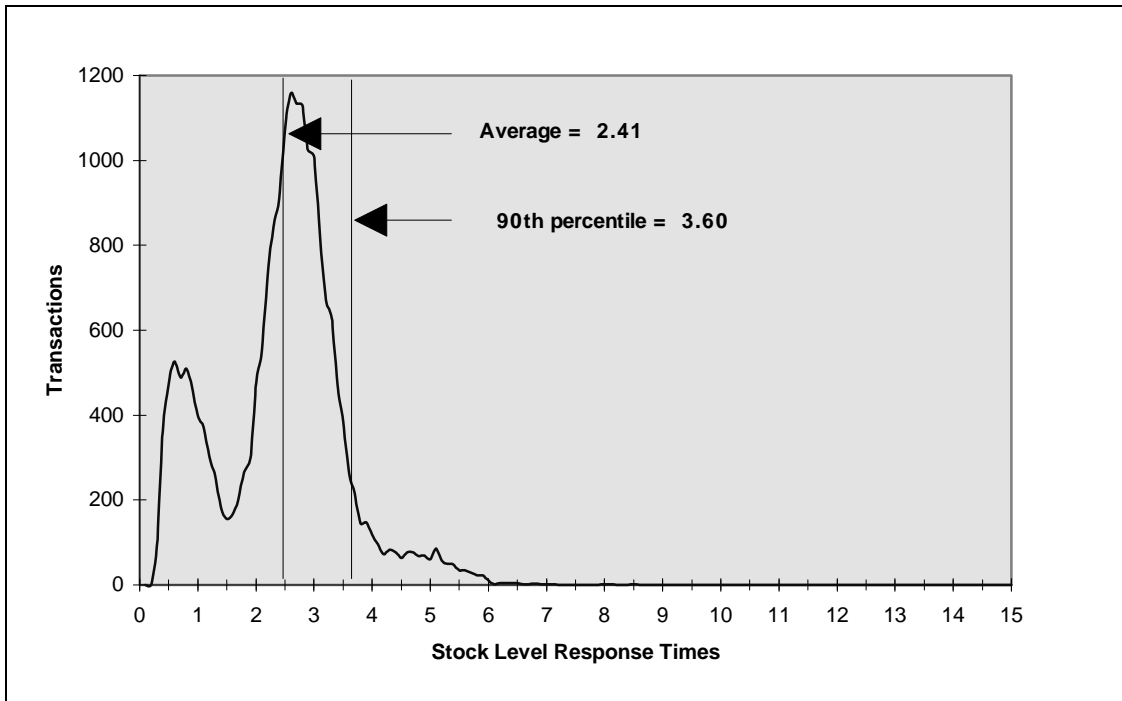
**Figure 5: Order Status Response Time Distribution**



**Figure 6: Delivery Response Time Distribution**



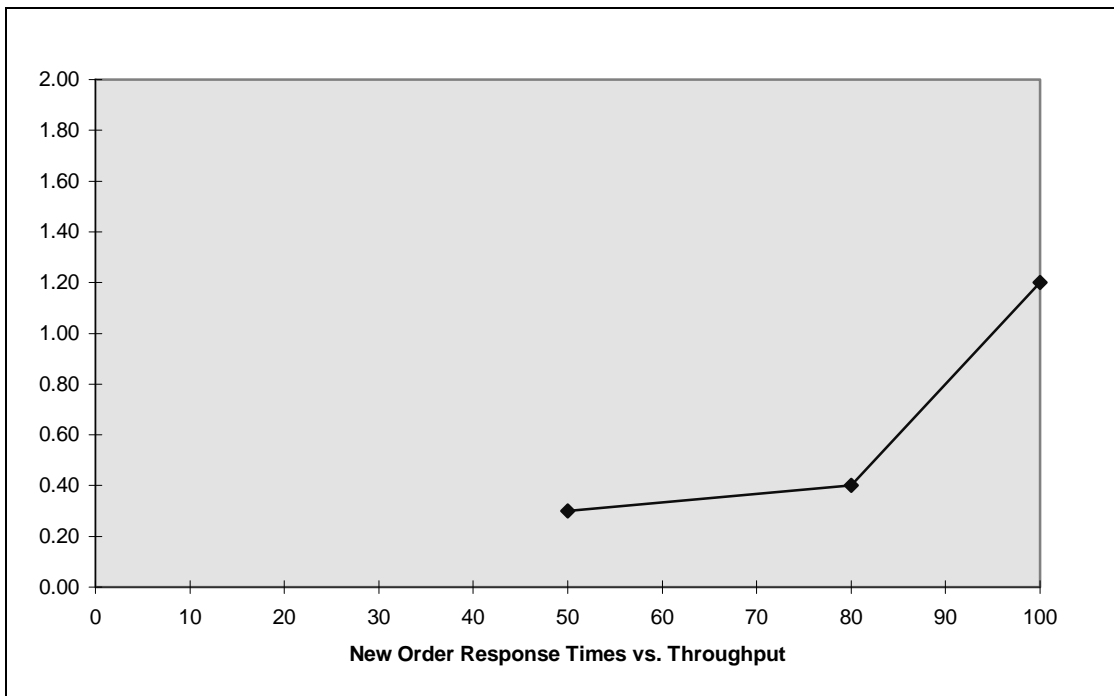
**Figure 7: Stock Level Response Time Distribution**



***New-Order Response Time vs. Throughput Graph***

*The performance curve for response times versus throughput (see Clause 5.6.2) must be reported for the New-Order transaction. (Clause 8.1.6.5)*

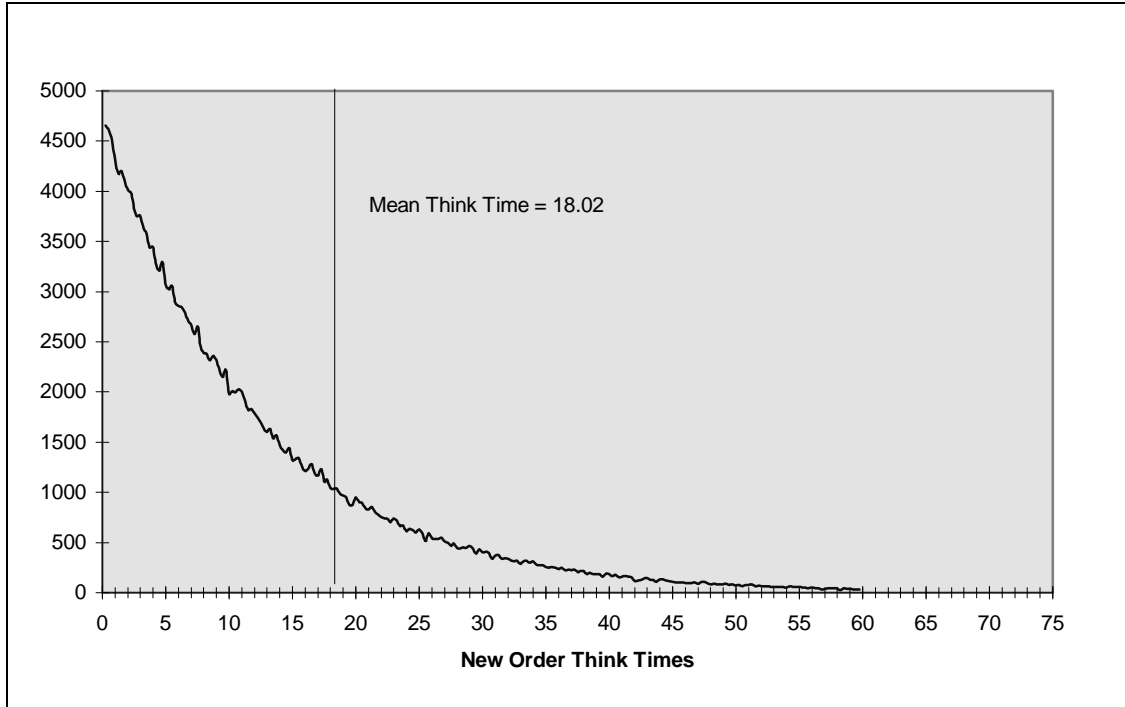
**Figure 8: New Order Response Time vs. Throughput**



## New-Order Think Time Distribution Graph

Think time frequency distribution curves (see Clause 5.6.3) must be reported for the New-Order transaction (Clause 8.1.66).

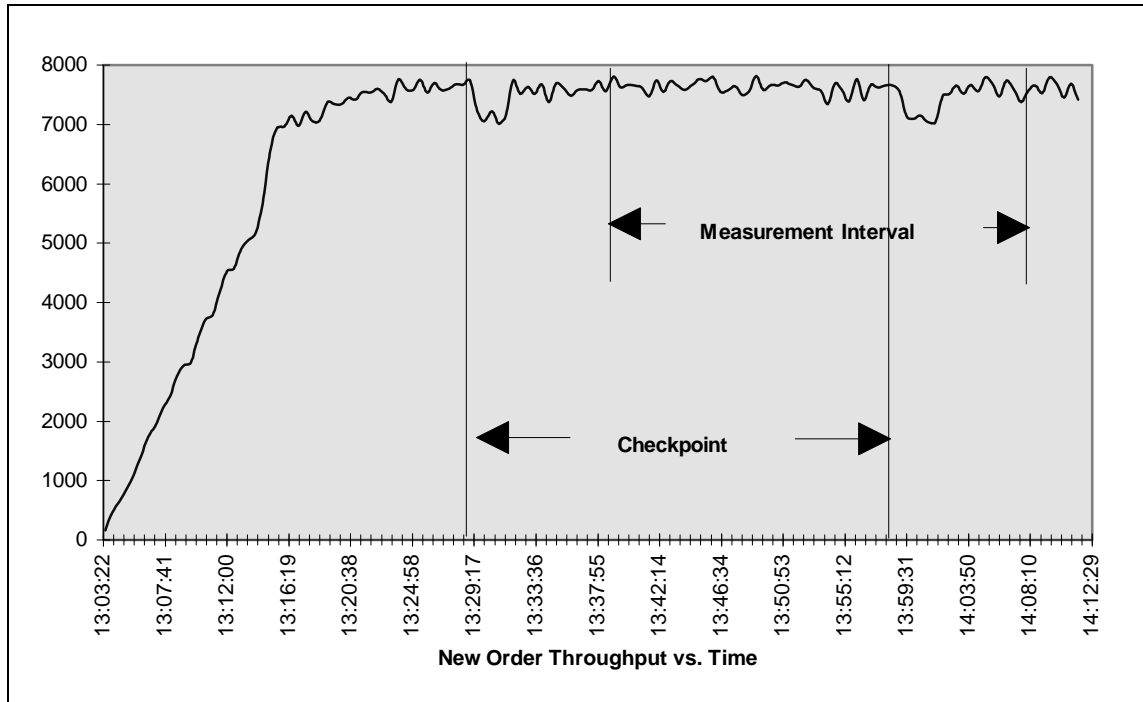
Figure 9: New Order Think Time Distribution



## Steady-State Graph

A graph of throughput versus elapsed time (see Clause 5.6.5) must be reported for the New-Order transaction (Clause 8.1.6.8).

Figure 10: 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 (Clause 8.1.6.9).

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

## 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 (Clause 8.1.6.10).

The RTE generated the required input data to choose a transaction from the menu. This data was timestamped. The response for the transaction was verified and timestamped in the RTE log files.

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 and was logged in the RTE log.

The RTE then waited the required think time interval before repeating the process starting at selecting another 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 another Ethernet LAN using Microsoft SQL Server DBLIB library and RPC calls.

To perform checkpoints at specific intervals, we set SQL Server *recovery interval* to the maximum allowable value and wrote a script to schedule multiple checkpoints at specific intervals. By setting the TRACE FLAG #3502, SQL Server logged the checkpoint beginning and ending time in the ERRORLOG file. The script included a wait time between each checkpoint equal to the measurement interval which was 30 minutes. The checkpoint script was started manually after the RTE had all users logged in and sending transactions.

At each checkpoint, Microsoft SQL Server wrote to disk all memory pages that had been updated but not yet physically written to disk. Upon completion of the checkpoint, Microsoft SQL Server wrote a special record to the recovery log to indicate that all disk operations had been satisfied to this point. The positioning of the checkpoint was verified to be clear of the guard zones and is depicted on the graph in Figure 10.

### ***Reproducibility Methodology***

*A description of the method used to determine the reproducibility of the measurement results must be reported (Clause 8.1.6.11).*

The duration of total time for the measurement(s) was sufficient to include three checkpoints, each thirty minutes apart from checkpoint completion to checkpoint start.. The stated throughput of the TPC-C benchmark, 7,573 tpmC, was obtained from the first half of the total running time. The reproducible portion was obtained from the second half of the total running time (e.g., it straddled the third and last checkpoint). The reported throughput was matched within 0.4%.

### ***Measurement Interval***

The measurement interval was 30 minutes.

## **Transaction Mix**

*The method of regulating the transaction mix (e.g., card decks or weighted random distribution) must be described. If weighted distribution is used and the RTE adjusts the weights associated with each transaction type, the maximum adjustments to the weight from the initial value must be disclosed (Clause 8.1.6.13).*

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

*The percentage of the total mix for each transaction type must be disclosed (Clause 8.1.6.14).*

**Table 9: Transaction Mix**

<b>Transaction</b>	<b>Percentage</b>
New Order	44.7
Payment	43.1
Order Status	4.08
Delivery	4.07
Stock Level	4.05

## **Checkpoints**

The initial checkpoint was started 26 minutes after the start of the ramp-up. The second checkpoint was started 30 minutes after the 1st checkpoint had completed. The checkpoint in the measurement interval lasted 2 minutes, 22 seconds. The measurement interval contains the second checkpoint, and is clear of the guard zones.

## Clause 6 -- SUT, Driver, and Communication Definition Related Items

---

### **RTE Parameters**

*The RTE input parameters, code fragments, functions, etc, used to generate each transaction input field must be disclosed (Clause 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 used was the Microsoft BenchCraft RTE System. The 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 (Clause 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) (Clause 8.1.7.3).*

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

Figures 1 & 2 of this report contain detailed diagrams of both the benchmark configuration and the priced configuration, and the associated text summarizes the differences between the two.

### **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). (Clause 8.1.7.4)*

All components of the priced configuration were operational during the benchmark. A high-speed switch was configured into the RTE-to-Client network topology to simplify connectivity. The switch was configured as 10baseT, half-duplex, which is the network capacity of the priced (and tested) 10baseT hub.

### **Network Bandwidth**

*The network bandwidth(s) used in the tested/priced configuration must be disclose. (Clause 8.1.7.5).*

The network bandwidth for priced and tested configurations was identical. The RTE-to-Client connections used 10baseT Ethernet, while the Client-to-Server connection used 100baseT Ethernet.

### **Operator Intervention**

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

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***

Hardware Availability Date: March 1, 1997  
Software Availability Date: March 31, 1997

### ***Measured TpmC***

Maximum Qualified Throughput: 7,573.00 TpmC  
Price Performance Metric: \$ 76.52 per TpmC

### ***Country Specific Pricing***

This system is priced for the United States of America.

### ***Usage Pricing***

The component pricing based on usage is shown below:

- Seven Microsoft Windows NT 4.0 Server licenses
- One (1) Microsoft SQL Server v.6.50 (Includes 5 Client Access Licenses).
- One (1) Microsoft Internet Connector License
- One (1) Microsoft SQL Server Programmers Toolkit
- One (1) Microsoft Visual C++ version 4.2
- 5 year replacement/repair warranty with spares on the Netlux and HiQ components.

### ***System Pricing***

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***

This TPC-C benchmark has been audited by Tom Sawyer of Performance Metrics.

Tom Sawyer of Performance Metrics, Inc.

### ***Availability of the Full Disclosure Report***

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

Transaction Processing Performance Council  
c/o Shanley Public Relations  
777 North First Street, Suite 6000  
San Jose, CA 95112-6311

or:

Amdahl Corporation  
1250 E. Arques Avenue  
Sunnyvale, CA 94088

Attention: John Howell

## Auditor Attestation Letter

March 21, 1997

Mr. John Howell  
Performance Analyst  
Amdahl Corporation  
1237 E. Arquez Avenue  
Sunnyvale CA 94088

Leah Schoeb  
Standard Benchmark Center Manager  
Intel Corporation  
2111 NE 25<sup>th</sup> Avenue  
Beaverton OR

I have verified the TPC Benchmark™ C client/server for the following configuration:

Platform: Amdahl EnVista Frontline Server™  
Database Manager: Microsoft SQL Server 6.5 plus Us  
Operating System: Microsoft Windows NT Server version 4.0

Server: Amdahl EnVista Frontline Server™				
CPU's	Memory	Disks	90% Response	tpmC
4 Pentium Pro's @ 200 MHz	Main: 2 GB Cache: 512 MB each	75 @ 4.3 GB 15 @ 9 GB	1.2 sec.	7,573.00
6 Clients: HiQ Personal Computer				
1 Pentium @ 133 MHz	Main: 128 MB Cache: 256 KB	1 @ 2 GB	na	na

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

- The transactions were correctly implemented.
- The database files were properly sized and populated.
- The database was properly scaled with 620 warehouses
- The ACID properties were met, including phantom protection.
- Input data was generated according to the specified percentages.
- Eight hours of mirrored log space was present on the tested system.
- The data for the 180 day calculation was verified.
- Fifteen 9GB drives were substituted for 4GB drives in the priced configuration. The substituted drives contained stripes of the same tables on 15 9GB drives in the configuration.
- The steady state portion of the test was 30 minutes.
- One checkpoint was taken before the measured interval.
- One checkpoint was taken during the measured interval.
- The checkpoints were verified to be clear of the guard zone.

- The system pricing was checked for major components and maintenance.
- Third party quotes were verified for compliance.

Sincerely,

A handwritten signature in cursive script that reads "Tom Sawyer". The signature is written in black ink and is positioned below the word "Sincerely,".

Tom Sawyer  
Auditor

# Appendix A - Application Source Code

---

## Appendix A - Application Source Code

### Microsoft WEB Client

#### Makefile

```
!IF "$(CFG)" == ""
CFG=Debug
!MESSAGE No configuration specified. Defaulting to Debug
!ENDIF

!IF "$(SQL_LOC)" == ""
SQL_LOC=C:\MSSQL\DLIB
!MESSAGE No SQL_LOC specified. Defaulting to C:\MSSQL\DLIB
!ENDIF

!IF "$(CFG)" != "Release" && "$(CFG)" != "Debug"
!MESSAGE Invalid configuration "$(CFG)" specified.
!MESSAGE You can specify a configuration when running NMAKE on this makefile
!MESSAGE by defining the macro CFG on the command line. For example:
!MESSAGE
!MESSAGE NMAKE CFG="Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "Release"
!MESSAGE "Debug"
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF

OUTDIR      = .
SRCDIR      = .\Src
OBJDIR      = .\Objs
OUTDIR      = .\Bin
DLIB        = $(SQL_LOC)
DLIBINC     = $(DLIB)\INCLUDE
DLIBDIR     = $(DLIB)\LIB

!IF "$(CFG)" != "Debug"
LDEBUG      =
CDEBUG      =
LDEBUG_RG   =
CDEBUG_RG   =
DEBUG       =
FLAGS       = /D "WIN32" /D "_WINDOWS"
OPT         = /Ot
!ELSE
LDEBUG      = /debug /pdb:$(OBJDIR)\tpcc.pdb
CDEBUG      = /Zi /Yd /Fd$(OBJDIR)\tpcc.pdb
LDEBUG_RG   = /debug /pdb:$(OBJDIR)\install.pdb
CDEBUG_RG   = /Zi /Yd /Fd$(OBJDIR)\install.pdb
FLAGS       = /D "_DEBUG" /D "WIN32" /D "_WINDOWS"
OPT         = /Od
!ENDIF

LINK32_LIBS = user32.lib msacm32.lib advapi32.lib $(DLIBDIR)\ntwdblib.lib
LINK32_OBJS = "$(OBJDIR)\tpcc.obj" "$(OBJDIR)\tpcc.res"
LINK32_DEF  = "$(SRCDIR)\tpcc.def"
LINK32_FLAGS = /nologo /subsystem:windows /dll /incremental:no $(LDEBUG) /def:"$(LINK32_DEF)" /out:"$(OBJDIR)\tpcc.dll"

LINK32_LIBS_RG = user32.lib gdi32.lib advapi32.lib version.lib comctl32.lib
LINK32_OBJS_RG = "$(OBJDIR)\install.obj" "$(OBJDIR)\install.res"
LINK32_FLAGS_RG = /nologo /subsystem:windows /incremental:no $(LDEBUG_RG) /out:$(OUTDIR)\install.exe

ALL: $(OBJDIR)\, $(OUTDIR)\, $(OUTDIR)\install.exe

$(OBJDIR).:
if not exist $(OBJDIR) md $(OBJDIR)

$(OUTDIR).:
if not exist $(OUTDIR) md $(OUTDIR)

"$(OBJDIR)\tpcc.obj": "$(SRCDIR)\tpcc.c" "$(SRCDIR)\tpcc.h"
cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I $(DLIBINC) $(FLAGS) /Fo$(OBJDIR)\tpcc.obj /c "$(SRCDIR)\tpcc.c"

$(OBJDIR)\tpcc.res: $(SRCDIR)\tpcc.rc
rc.exe /I 0x409 /fo $(OBJDIR)\tpcc.res $(FLAGS) $(SRCDIR)\tpcc.rc

$(OBJDIR)\tpcc.dll: $(LINK32_OBJS) $(LINK32_DEF)
link.exe $(LINK32_FLAGS) $(LINK32_OBJS) $(LINK32_LIBS)

$(OBJDIR)\delisrv.exe: $(SRCDIR)\delisrv.c $(SRCDIR)\delisrv.h
cl.exe /nologo /MT /W3 $(CDEBUG) $(OPT) /I $(DLIBINC) $(FLAGS) /Fo$(OBJDIR)\delisrv.obj $(SRCDIR)\delisrv.c /link /out:$(OBJDIR)\delisrv.exe
$(DLIBDIR)\ntwdblib.lib msacm32.lib advapi32.lib
```

# Appendix A - Application Source Code

```
$(OBJDIR)\install.res: $(SRCDIR)\install.rc $(OBJDIR)\tpcc.dll $(OBJDIR)\delisrv.exe
rc.exe /I 0x409 /fo$(OBJDIR)\install.res /i $(OBJDIR) /i $(SRCDIR) $(FLAGS) $(SRCDIR)\install.rc
```

```
$(OBJDIR)\install.obj: $(SRCDIR)\install.c $(OBJDIR)\tpcc.dll $(OBJDIR)\delisrv.exe $(OBJDIR)\install.res
cl -W3 $(CDEBUG_RG) /Fo$(OBJDIR)\install.obj /c $(SRCDIR)\install.c
```

```
$(OUTDIR)\install.exe: $(OBJDIR)\install.obj $(OBJDIR)\install.res
link.exe @<<
$(LINK32_FLAGS_RG) $(LINK32_OBJS_RG) $(LINK32_LIBS_RG)
<<
```

## Delisrv.h

```
/* FILE: DELISRV.H
 * Microsoft TPC-C Kit Ver. 3.00.000
 * Audited 08/23/96, By Francois Raab
 *
 * Copyright Microsoft, 1996
 *
 * PURPOSE: Header file for delivery service executable
 * Author: Philip Durr
 * philipdu@Microsoft.com
 */

#define AVAILABLE 0 //queue array element available
#define WRITE_LOCKED 1 //queue array element is being written to
#define READ_LOCKED 2 //queue array element is begin read
#define INUSE 4 //queue array element has
information stored in it

#define CTRL_C 3 //<Ctrl> C, exit key code

#define DEFCLPACKSIZE 4096 //default DB Library SQL Connection pack size

#define ERR_SUCCESS 0 //Success, no error.
#define ERR_CANNOT_CREATE_THREAD 1000 //Cannot create thread.
#define ERR_DBGETDATA_FAILED 1001 //Get data failed.
#define ERR_REGISTRY_NOT_SETUP 1002 //Registry not setup for tpcc.
#define ERR_CANNOT_ACCESS_DELIVERY_FN 1003 //Cannot access ReadDelivery cache.
#define ERR_CANNOT_ACCESS_REGISTRY 1004 //Cannot access registry key TPCC.
#define ERR_CANNOT_CREATE_RESULTS_FILE 1005 //Cannot create results file.
#define ERR_CANNOT_OPEN_PIPE 1006 //Cannot open delivery pipe.
#define ERR_READ_PIPE 1007 //Error reading pipe
#define ERR_INSUFFICIENT_MEMORY 1008 //insufficient memory

typedef struct _DELIVERY_TRANSACTION
{
    SYSTEMTIME queue; //time delivery transaction queued
    short w_id; //delivery warehouse
    short o_carrier_id; //carrier id
} DELIVERY_TRANSACTION;

typedef DELIVERY_TRANSACTION *LPDELIVERY_TRANSACTION; //pointer to delivery transaction queue

typedef struct _DELIVERY_PACKET
{
    BOOL bInUse; //entry current in use
    OVERLAPPED ov; //pipe io overlapped structure
    DELIVERY_TRANSACTION trans; //delivery transaction information
} DELIVERY_PACKET, *LPDELIVERY_PACKET;

typedef struct _SERRORMSG
{
    int iError; //error message id
    char szMsg[80]; //error message
} SERRORMSG;

//delivery transaction structure
typedef struct DELIVERY
{
    short w_id; //warehouse id
    short o_carrier_id; //carrier id
    int spid; //db library spid
    long o_id[10]; //returned delivery transaction ids
    DBPROCESS *dbproc; //db library DBPROCESS pointer
    SYSTEMTIME queue; //delivery transaction queue time
    SYSTEMTIME trans_end; //delivery transaction finished time
} DELIVERY;

typedef DELIVERY *LPDELIVERY; //pointer to delivery structure

//function prototypes
void main(int argc, char *argv[]);
static void cls(void);
static int RunDelivery(void);
static void QuitStatus(void);
static void AnimateWait1(void);
static void AnimateWait(void);
static int Init(void);
```

## Appendix A - Application Source Code

---

```
static void Restore(void);
static void ErrorMessage(int iError);
static BOOL GetParameters(int argc, char *argv[]);
static void PrintParameters(void);
static void PrintHeader(void);
static int ReadRegistrySettings(void);
static void CheckKey(void *ptr);
static void DeliveryHandler( void *ptr );
static void DeliveryThread( void *ptr );
static int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr, char *dberrstr, char *oserrstr);
static int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int severity, char *msgtext);
static BOOL SQLOpenConnection(DBPROCESS **dbproc, char *server, char *database, char *user, char *password, int *spid);
static void WriteLog(LPDELIVERY pDelivery);
static void CalculateElapsedTime(int *pElapsed, LPSYSTEMTIME lpBegin, LPSYSTEMTIME lpEnd);
static int SQLDelivery(DELIVERY *pDelivery);
static BOOL SQLDetectDeadlock(DBPROCESS *dbproc);
static BOOL ReadDeliveryInfo(short *w_id, short *o_carrier_id);
static BOOL PostDeliveryInfo(short w_id, short o_carrier_id);
static int OpenLogFile(void);
```

### Httpext.h

```
/*
 * Copyright (c) 1995 Process Software Corporation
 * Copyright (c) 1995 Microsoft Corporation
 *
 * Module Name : HttpExt.h
 *
 * Abstract :
 *
 * This module contains the structure definitions and prototypes for the
 * version 1.0 HTTP Server Extension interface.
 */
*****/

#ifndef _HTTPEXT_H_
#define _HTTPEXT_H_

#include <windows.h>

#ifdef __cplusplus
extern "C" {
#endif

#define HSE_VERSION_MAJOR 1 // major version of this spec
#define HSE_VERSION_MINOR 0 // minor version of this spec
#define HSE_LOG_BUFFER_LEN 80
#define HSE_MAX_EXT_DLL_NAME_LEN 256

typedef LPVOID HCONN;

// the following are the status codes returned by the Extension DLL

#define HSE_STATUS_SUCCESS 1
#define HSE_STATUS_SUCCESS_AND_KEEP_CONN 2
#define HSE_STATUS_PENDING 3
#define HSE_STATUS_ERROR 4

// The following are the values to request services with the ServerSupportFunction.
// Values from 0 to 1000 are reserved for future versions of the interface

#define HSE_REQ_BASE 0
#define HSE_REQ_SEND_URL_REDIRECT_RESP (HSE_REQ_BASE + 1)
#define HSE_REQ_SEND_URL (HSE_REQ_BASE + 2)
#define HSE_REQ_SEND_RESPONSE_HEADER (HSE_REQ_BASE + 3)
#define HSE_REQ_DONE_WITH_SESSION (HSE_REQ_BASE + 4)
#define HSE_REQ_END_RESERVED 1000

//
// These are Microsoft specific extensions
//

#define HSE_REQ_MAP_URL_TO_PATH (HSE_REQ_END_RESERVED+1)
#define HSE_REQ_GET_SSPI_INFO (HSE_REQ_END_RESERVED+2)

//
// passed to GetExtensionVersion
//

typedef struct _HSE_VERSION_INFO {
    DWORD dwExtensionVersion;
    CHAR lpszExtensionDesc[HSE_MAX_EXT_DLL_NAME_LEN];
} HSE_VERSION_INFO, *LPHSE_VERSION_INFO;
```

# Appendix A - Application Source Code

---

```
//
// passed to extension procedure on a new request
//
typedef struct _EXTENSION_CONTROL_BLOCK {

    DWORD   cbSize;           // size of this struct.
    DWORD   dwVersion;       // version info of this spec
    HCONN   ConnID;          // Context number not to be modified!
    DWORD   dwHttpStatusCode; // HTTP Status code
    CHAR    lpszLogData[HSE_LOG_BUFFER_LEN]; // null terminated log info specific to this Extension DLL

    LPSTR   lpszMethod;      // REQUEST_METHOD
    LPSTR   lpszQueryString; // QUERY_STRING
    LPSTR   lpszPathInfo;    // PATH_INFO
    LPSTR   lpszPathTranslated; // PATH_TRANSLATED

    DWORD   cbTotalBytes;    // Total bytes indicated from client
    DWORD   cbAvailable;    // Available number of bytes
    LPBYTE  lpbData;        // pointer to cbAvailable bytes

    LPSTR   lpszContentType; // Content type of client data

    BOOL (WINAPI * GetServerVariable) ( HCONN   hConn,
                                        LPSTR   lpszVariableName,

                                        LPVOID  lpvBuffer,

                                        LPDWORD  lpdwSize );

    BOOL (WINAPI * WriteClient) ( HCONN   ConnID,
                                LPVOID  Buffer,
                                LPDWORD  lpdwBytes,
                                DWORD    dwReserved );

    BOOL (WINAPI * ReadClient) ( HCONN   ConnID,
                                LPVOID  lpvBuffer,
                                LPDWORD  lpdwSize );

    BOOL (WINAPI * ServerSupportFunction)( HCONN   hConn,
                                           DWORD   dwHSERRequest,
                                           LPVOID  lpvBuffer,
                                           LPDWORD  lpdwSize,
                                           LPDWORD  lpdwDataType );

} EXTENSION_CONTROL_BLOCK, *LPEXTENSION_CONTROL_BLOCK;

//
// these are the prototypes that must be exported from the extension DLL
//
BOOL WINAPI GetExtensionVersion( HSE_VERSION_INFO *pVer );
DWORD WINAPI HttpExtensionProc( EXTENSION_CONTROL_BLOCK *pECB );

// the following type declarations is for the server side

typedef BOOL (WINAPI * PFN_GETEXTENSIONVERSION)( HSE_VERSION_INFO *pVer );
typedef DWORD (WINAPI * PFN_HTTPEXTENSIONPROC)( EXTENSION_CONTROL_BLOCK *pECB );

#ifdef __cplusplus
}
#endif

#endif // end definition_HTTPEXT_H_
```

## Resource.h

```
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by TPCC.rc
//
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifdef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE 101
#define _APS_NEXT_COMMAND_VALUE 40001
#define _APS_NEXT_CONTROL_VALUE 1000
#define _APS_NEXT_SYMED_VALUE 101
#endif
#endif
```

## Tpcc.h

```
/* FILE: TPCC.H
 * Microsoft TPC-C Kit Ver. 3.00.000
 * Audited 08/23/96, By Francois Raab
 *
 * Copyright Microsoft, 1996
 *
```



# Appendix A - Application Source Code

```

*      PURPOSE:  Header file for ISAPI TPCC.DLL, defines structures and functions used in the isapi tpcc.dll.
*      Author:   Philip Durr
*               philipdu@Microsoft.com
*/

//VERSION RESOURCE DEFINES
#define _APS_NEXT_RESOURCE_VALUE          101
#define _APS_NEXT_COMMAND_VALUE          4001
#define _APS_NEXT_CONTROL_VALUE          1000
#define _APS_NEXT_SYMED_VALUE            101

#define ERR_TYPE_WEBDLL                    1
#define ERR_TYPE_SQL                       2
#define ERR_TYPE_DBLIB                     3

#define ERR_SUCCESS                          1000      //Success, no
error.
#define ERR_COMMAND_UNDEFINED                1001      //Command undefined.
#define ERR_NOT_IMPLEMENTED_YET              1002      //Not Implemented Yet.
#define ERR_CANNOT_INIT_TERMINAL             1003      //Cannot initialize client connection.
#define ERR_OUT_OF_MEMORY                    1004      //insufficient memory.
#define ERR_NEW_ORDER_NOT_PROCESSED          1005      //Cannot process new Order form.
#define ERR_PAYMENT_NOT_PROCESSED           1006      //Cannot process payment form.
#define ERR_NO_SERVER_SPECIFIED             1007      //No Server name specified.
#define ERR_ORDER_STATUS_NOT_PROCESSED      1008      //Cannot process order status form.
#define ERR_W_ID_INVALID                     1009      //Invalid Warehouse ID.
#define ERR_CAN_NOT_SET_MAX_CONNECTIONS     1010      //Insufficient memory to allocate # connections.
#define ERR_NOSUCH_CUSTOMER                 1011      //No such customer.
#define ERR_D_ID_INVALID                    1012      //Invalid District ID Must be 1 to 10.
#define ERR_MAX_CONNECT_PARAM               1013      //Max client connections exceeded, run
install to increase.
#define ERR_INVALID_SYNC_CONNECTION         1014      //Invalid Terminal Sync ID.
#define ERR_INVALID_TERMID                  1015      //Invalid Terminal ID.
#define ERR_PAYMENT_INVALID_CUSTOMER        1016      //Payment Form, No such Customer.
#define ERR_SQL_OPEN_CONNECTION             1017      //SQLOpenConnection API Failed.
#define ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY 1018      //Stock Level missing Threshold key "TT*".
#define ERR_STOCKLEVEL_THRESHOLD_INVALID    1019      //Stock Level Threshold invalid data type range = 1 - 99.
#define ERR_STOCKLEVEL_THRESHOLD_RANGE      1020      //Stock Level Threshold out of range, range must be 1 -
99.
#define ERR_STOCKLEVEL_NOT_PROCESSED        1021      //Stock Level not processed.
#define ERR_NEWORDER_FORM_MISSING_DID       1022      //New Order missing District key "DID*".
#define ERR_NEWORDER_DISTRICT_INVALID       1023      //New Order District ID Invalid range 1 - 10.
#define ERR_NEWORDER_DISTRICT_RANGE         1024      //New Order District ID out of Range. Range = 1 - 10.
#define ERR_NEWORDER_CUSTOMER_KEY          1025      //New Order missing Customer key "CID*".
#define ERR_NEWORDER_CUSTOMER_INVALID       1026      //New Order customer id invalid data type, range = 1 to
3000.
#define ERR_NEWORDER_CUSTOMER_RANGE         1027      //New Order customer id out of range, range = 1 to 3000.
#define ERR_NEWORDER_MISSING_IID_KEY        1028      //New Order missing Item Id key "IID*".
#define ERR_NEWORDER_ITEM_BLANK_LINES       1029      //New Order blank order lines all orders must be
continuous.
#define ERR_NEWORDER_ITEMID_INVALID         1030      //New Order Item Id is wrong data type, must be numeric.
#define ERR_NEWORDER_MISSING_SUPPW_KEY      1031      //New Order missing Supp_W key "SP##*".
#define ERR_NEWORDER_SUPPW_INVALID          1032      //New Order Supp_W invalid data type must be numeric.
#define ERR_NEWORDER_MISSING_QTY_KEY        1033      //New Order Missing Qty key "Qty##*".
#define ERR_NEWORDER_QTY_INVALID           1034      //New Order Qty invalid must be numeric range 1 - 99.
#define ERR_NEWORDER_SUPPW_RANGE            1035      //New Order Supp_W value out of range range = 1 - Max
Warehouses.
#define ERR_NEWORDER_ITEMID_RANGE           1036      //New Order Item Id is out of range. Range = 1 to 999999.
#define ERR_NEWORDER_QTY_RANGE              1037      //New Order Qty is out of range. Range = 1
to 99.
#define ERR_PAYMENT_DISTRICT_INVALID         1038      //Payment District ID is invalid must be 1 - 10.
#define ERR_NEWORDER_SUPPW_WITHOUT_ITEMID   1039      //New Order Supp_W field entered without a corresponding Item_Id.
#define ERR_NEWORDER_QTY_WITHOUT_ITEMID     1040      //New Order Qty entered without a corresponding Item_Id.
#define ERR_NEWORDER_NOITEMS_ENTERED        1041      //New Order Blank Items between items, items must be
continuous.
#define ERR_PAYMENT_MISSING_DID_KEY         1042      //Payment missing District Key "DID*".
#define ERR_PAYMENT_DISTRICT_RANGE          1043      //Payment District Out of range, range = 1 - 10.
#define ERR_PAYMENT_MISSING_CID_KEY         1044      //Payment missing Customer Key "CID*".
#define ERR_PAYMENT_CUSTOMER_INVALID        1045      //Payment Customer data type invalid, must be numeric.
#define ERR_PAYMENT_MISSING_CLT              1046      //Payment missing Customer LastName
Key "CLT*".
#define ERR_PAYMENT_LAST_NAME_TO_LONG       1047      //Payment Customer last name longer than 16 characters.
#define ERR_PAYMENT_CUSTOMER_RANGE          1048      //Payment Customer ID out of range, must be 1 to 3000.
#define ERR_PAYMENT_CID_AND_CLT             1049      //Payment Customer ID and Last Name
entered must be one or other.
#define ERR_PAYMENT_MISSING_CDI_KEY         1050      //Payment missing Customer district key "CDI*".
#define ERR_PAYMENT_CDI_INVALID             1051      //Payment Customer district invalid must be
numeric.
#define ERR_PAYMENT_CDI_RANGE               1052      //Payment Customer district out of range
must be 1 - 10.
#define ERR_PAYMENT_MISSING_CWI_KEY         1053      //Payment missing Customer Warehouse key "CWI*".
#define ERR_PAYMENT_CWI_INVALID             1054      //Payment Customer Warehouse invalid
must be numeric.
#define ERR_PAYMENT_CWI_RANGE               1055      //Payment Customer Warehouse out of
range, 1 to Max Warehouses.
#define ERR_PAYMENT_MISSING_HAM_KEY         1056      //Payment missing Amount key "HAM*".
#define ERR_PAYMENT_HAM_INVALID            1057      //Payment Amount invalid data type must
be numeric.
#define ERR_PAYMENT_HAM_RANGE               1058      //Payment Amount out of range, 0 -
9999.99.
#define ERR_ORDERSTATUS_MISSING_DID_KEY     1059      //Order Status missing District key "DID*".

```

## Appendix A - Application Source Code

```

#define ERR_ORDERSTATUS_DID_INVALID 1060 //Order Status District invalid, value must be numeric 1 -
10.
#define ERR_ORDERSTATUS_DID_RANGE 1061 //Order Status District out of range must be 1 - 10.
#define ERR_ORDERSTATUS_MISSING_CID_KEY 1062 //Order Status missing Customer key "CID*".
#define ERR_ORDERSTATUS_MISSING_CLT_KEY 1063 //Order Status missing Customer Last Name key "CLT*".
#define ERR_ORDERSTATUS_CLT_RANGE 1064 //Order Status Customer last name longer than 16
characters.
#define ERR_ORDERSTATUS_CID_INVALID 1065 //Order Status Customer ID invalid, range must be numeric
1 - 3000.
#define ERR_ORDERSTATUS_CID_RANGE 1066 //Order Status Customer ID out of range must be 1 - 3000.
#define ERR_ORDERSTATUS_CID_AND_CLT 1067 //Order Status Customer ID and LastName entered must
be only one."
#define ERR_DELIVERY_MISSING_OCD_KEY 1068 //Delivery missing Carrier ID key "\OCD*".
#define ERR_DELIVERY_CARRIER_INVALID 1069 //Delivery Carrier ID invalid must be numeric 1 - 10.
#define ERR_DELIVERY_CARRIER_ID_RANGE 1070 //Delivery Carrier ID out of range must be 1 - 10.
#define ERR_PAYMENT_MISSING_CLT_KEY 1071 //Payment missing Customer Last Name key "CLT*".

//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 is used for posting delivery transactions
typedef struct _DELIVERY_TRANSACTION
{
    SYSTEMTIME queue; //time delivery transaction queued
    short w_id; //delivery warehouse
    short o_carrier_id; //carrier id
} DELIVERY_TRANSACTION;

//error message structure used in ErrorMessage API
typedef struct _SERRORMSG
{
    int iError; //error id of message
    char szMsg[80]; //message to sent to browser
} SERRORMSG;

//This structure defines the data necessary to keep distinct for each terminal or client connection.
typedef struct _CLIENTDATA
{
    int inUse; //in use flag allows client entries to be reused
    int w_id; //warehouse id assigned at welcome form
    int d_id; //district id assigned at welcome form

    DBPROCESS *dbproc; //dblib connection pointer
    int spid; //spid assigned from dblib
    int iSyncId; //synchronization id
    int iTickCount; //time of last access;
    int iTermId; //terminal id of http stream connection

    char szBuffer[4096]; //form buffer each HTML form is built for a client in here

    NEW_ORDER_DATA newOrderData; //new order form data
    PAYMENT_DATA paymentData; //payment form data
    ORDER_STATUS_DATA orderStatusData; //order status form data
    DELIVERY_DATA deliveryData; //delivery form data
    STOCK_LEVEL_DATA stockLevelData; //stock level form data
} CLIENTDATA;

typedef CLIENTDATA *PCLIENTDATA; //pointer to client structure

//This structure is used to define the operational interface for terminal id support
typedef struct _TERM
{
    int iAvailable; //total allocated terminal array
    int iNext; //next available
    terminal array element
    int iMasterSyncId; //synchronization id
    flag
    BOOL bInit; //structure has been initialized
    CLIENTDATA *pClientData; //pointer to allocated client data
    void (*Init)(void); //API to initialize this structure
    int (*Allocate)(void); //API to allocate a new terminal entry array
    id returned
    void (*Restore)(void); //API to free terminal data
    int (*Add)(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString); //API to add a terminal id to
array, this context will

    //be passed from the browser to the tpc.dll in the
    //TERMID= key in the HTTP string.

```

# Appendix A - Application Source Code

```
void          (*Delete)(EXTENSION_CONTROL_BLOCK *pECB, int id);          //API to free resources used by a terminal array entry
} TERM;

typedef TERM *PTERM;                                                    //pointer to terminal structure type

#ifdef USE_ODBC
typedef struct _DBPROCESS
{
    HDDBC          hdbc;
    HSTMT          hstmt;
    int            spid;
    int            iTermId;
    int            iSyncId;
    BOOL          bDeadlock;
    EXTENSION_CONTROL_BLOCK *pECB;
} DBPROCESS, *PDBPROCESS;
#else
//this structure allows the EXTENSION CONTROL BLOCK to be passed to the msg and error handlers.
typedef struct _ECBINFO
{
    int            iTermId;          //terminal id
    int            iSyncId;        //browser sync id
    BOOL          bDeadlock;       //deadlock condition flag
    BOOL          bFailed;        //cleared before sql transaction, set in err handlers if an error
} ECBINFO, *PECBINFO;

occurs          EXTENSION_CONTROL_BLOCK *pECB;          //inetsrv current connection structure information
} ECBINFO, *PECBINFO;
#endif

//function prototypes
BOOL WINAPI DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved);
static void DeliveryDisconnect(void *ptr);
static BOOL IsValidTermId(int TermId);
BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB, int *pCmd, int *pFormId, int *pTermId, int *pSyncId);
void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId);
static void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr);
static void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format, ...);
void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int iErrorType, char *szMsg, int iTermId, int iSyncId);
static BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue, int iMax);
static void TermInit(void);
int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr, char *dberrstr, char *oserrstr);
int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int severity, char *msgtext);
static void TermRestore(void);
static int TermAllocate(void);
static int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString);
static void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int id);
BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, char *szServer, char *szUser, char *szPassword, char *szDatabase);
static BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS **dbproc, char *server, char *database, char *user, char *password, char *app, int *spid);
static BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS *dbproc);
static BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short deadlock_retry);
static int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, NEW_ORDER_DATA *pNewOrder, short deadlock_retry);
static int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short deadlock_retry);
static int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short deadlock_retry);
BOOL SQLDetectDeadlock(DBPROCESS *dbproc);
static void FormatString(char *szDest, char *szPic, char *szSrc);
static char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeMainMenuForm(int iTermId, int iSyncId);
static char *MakeWelcomeForm(void);
static char *MakeNewOrderForm(int iTermId, int iSyncId, BOOL bInput, BOOL bValid);
static char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL bInput);
static char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput);
static void UtilStrCpy(char *pDest, char *pSrc, int n);
static void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
static int GetNewOrderData(LPSTR lpszQueryString, NEW_ORDER_DATA *pNewOrderData);
static int GetPaymentData(LPSTR lpszQueryString, PAYMENT_DATA *pPaymentData);
static int GetOrderStatusData(LPSTR lpszQueryString, ORDER_STATUS_DATA *pOrderStatusData);
static BOOL ReadRegistrySettings(void);
static BOOL PostDeliveryInfo(short w_id, short o_carrier_id);
static BOOL IsNumeric(char *ptr);
static int ODBCError(DBPROCESS *dbproc);
static void FormatHTMLString(char *szBuff, char *szStr, int iLen);
```

# Appendix A - Application Source Code

---

## Trans.h

```
/*      FILE:                TRANS.H
 *
 *      Microsoft TPC-C Kit Ver. 3.00.000
 *      Audited 08/23/96      By Francois Raab
 *
 *      PURPOSE:  Header file for ISAPI TPCC.DLL, defines structures and functions used in the isapi tpcc.dll.
 *
 *
 *      Copyright Microsoft inc. 1996, All Rights Reserved
 *
 *
 *      Author:              PhilipDu, from tpcc.h by DamienL
 *                          DamienL@Microsoft.com
 *                          philipdu@Microsoft.com
 */

#ifndef _INC_TRANS

#define _INC_TRANS

#ifdef USE_ODBC
#ifdef TIMESTAMP_STRUCT
#include <sqltypes.h>
#endif
#else
#ifdef _INC_SQLFRONT
#include <sqlfront.h>
#endif
#endif

#ifdef DBINT
typedef long DBINT;
#endif

#define DEFCLPACKSIZE 4096
#define DEADLOCKWAIT 10

// String length constants
#define SERVER_NAME_LEN 20
#define DATABASE_NAME_LEN 20
#define USER_NAME_LEN 20
#define PASSWORD_LEN 20
#define TABLE_NAME_LEN 20
#define I_DATA_LEN 50
#define I_NAME_LEN 24
#define BRAND_LEN 1
#define LAST_NAME_LEN 16
#define W_NAME_LEN 10
#define ADDRESS_LEN 20
#define STATE_LEN 2
#define ZIP_LEN 9
#define S_DIST_LEN 24
#define S_DATA_LEN 50
#define D_NAME_LEN 10
#define FIRST_NAME_LEN 16
#define MIDDLE_NAME_LEN 2
#define PHONE_LEN 16
#define DATETIME_LEN 30
#define CREDIT_LEN 2
#define C_DATA_LEN 250
#define H_DATA_LEN 24
#define DIST_INFO_LEN 24
#define MAX_OL_NEW_ORDER_ITEMS 15
#define MAX_OL_ORDER_STATUS_ITEMS 15
#define STATUS_LEN 25
#define OL_DIST_INFO_LEN 24

// transaction structures

typedef struct
{
    short ol_supply_w_id;
    long ol_i_id;
    char ol_i_name[I_NAME_LEN+1];
    short ol_quantity;
    char ol_brand_generic[BRAND_LEN+1];
    double ol_i_price;
    double ol_amount;
    short ol_stock;
    short num_warehouses;
} OL_NEW_ORDER_DATA;

typedef struct
{
    short w_id;
    short d_id;
    long c_id;
    short o_ol_cnt;
    char c_last[LAST_NAME_LEN+1];
    char c_credit[CREDIT_LEN+1];
    double c_discount;
    double w_tax;
    double d_tax;
}
```

## Appendix A - Application Source Code

---

```
        long                o_id;
        short               o_commit_flag;
#ifdef USE_ODBC
        TIMESTAMP_STRUCT    o_entry_d;
#else
        DBDATERECS         o_entry_d;
#endif
        short              o_all_local;
        double             total_amount;
        long               num_deadlocks;
        char               execution_status[STATUS_LEN];
        OL_NEW_ORDER_DATA OI[MAX_OL_NEW_ORDER_ITEMS];
} NEW_ORDER_DATA;

typedef struct
{
        short              w_id;
        short              d_id;
        long               c_id;
        short              c_d_id;
        short              c_w_id;
        double             h_amount;
#ifdef USE_ODBC
        TIMESTAMP_STRUCT    h_date;
#else
        DBDATERECS         h_date;
#endif
        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_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];
#ifdef USE_ODBC
        TIMESTAMP_STRUCT    c_since;
#else
        DBDATERECS         c_since;
#endif
        char               c_credit[CREDIT_LEN+1];
        double             c_credit_lim;
        double             c_discount;
        double             c_balance;
        char               c_data[200+1];
        long               num_deadlocks;
        char               execution_status[STATUS_LEN];
} PAYMENT_DATA;

typedef struct
{
        long                ol_i_id;
        short               ol_supply_w_id;
        short               ol_quantity;
        double              ol_amount;
#ifdef USE_ODBC
        TIMESTAMP_STRUCT    ol_delivery_d;
#else
        DBDATERECS         ol_delivery_d;
#endif
} OL_ORDER_STATUS_DATA;

typedef struct
{
        short              w_id;
        short              d_id;
        long               c_id;
        char               c_first[FIRST_NAME_LEN+1];
        char               c_middle[MIDDLE_NAME_LEN+1];
        char               c_last[LAST_NAME_LEN+1];
        double             c_balance;
        long               o_id;
#ifdef USE_ODBC
        TIMESTAMP_STRUCT    o_entry_d;
#else
        DBDATERECS         o_entry_d;
#endif
        short              o_carrier_id;
        OL_ORDER_STATUS_DATA OIOrderStatusData[MAX_OL_ORDER_STATUS_ITEMS];
        short              o_ol_cnt;
```

## Appendix A - Application Source Code

---

```
        long        num_deadlocks;
        char        execution_status[STATUS_LEN];
    } ORDER_STATUS_DATA;

typedef struct
{
    long            o_id;
} DEL_ITEM;

typedef struct
{
    short          w_id;
    short          o_carrier_id;
    SYSTEMTIME     queue_time;
    long          num_deadlocks;
    DEL_ITEM      DelItems[10];
    char          execution_status[STATUS_LEN];
} DELIVERY_DATA;

typedef struct
{
    short          w_id;
    short          d_id;
    short          thresh_hold;
    long          low_stock;
    long          num_deadlocks;
    char          execution_status[STATUS_LEN];
} STOCK_LEVEL_DATA;

#endif
```

### Delisrv.c

```
/*      FILE:          DELISRV.C
 *
 *      Microsoft TPC-C Kit Ver. 3.00.000
 *      Audited 08/23/96, By Francois Raab
 *
 *      Copyright Microsoft, 1996
 *
 *      PURPOSE:      Delivery TPC-C transaction executable
 *      Author:       Philip Durr
 *
 *      philipdu@Microsoft.com
 */

#include <windows.h>
#include <process.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 <conio.h>
#include <ctype.h>

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

#include "delisrv.h"

char        szServer[32];           //SQL server name
char        szDatabase[32];       //tpcc database name
char        szUser[32];           //user name
char        szPassword[32];       //user password
int         iNumThreads           = 4;           //number of threads to create
int         iDelayMs              = 1000;       //delay between delivery
queue checks
int         iDeadlockRetry        = 3;         //number of read check
retries.
int         iQSlotts              = 3000;      //delivery transaction queues

FILE        *fpLog;                //pointer to log
file
CRITICAL_SECTION WriteLogCriticalSection; //critical section for delivery write log
CRITICAL_SECTION DeliveryCriticalSection; //critical section for delivery transactions cache
static      LPTSTR      lpszPipeName = TEXT("\\\\.\\pipe\\DELISRV"); //delivery pipe name

HANDLE      hPipe                = INVALID_HANDLE_VALUE; //delivery pipe handle
HANDLE      hComPort              = INVALID_HANDLE_VALUE; //delivery pipe completion port handle.

BOOL        bDone;
//delivery executable termination request flag
BOOL        bFlush;
//Flush delivery log info when written.

LPDELIVERY_PACKET      pDeliveryCache;
```

## Appendix A - Application Source Code

---

```
int                                     versionMS = 3;
//delivery executable version number.
int                                     versionMM = 0;
//formatted as MS.MM.LS, 1.00.005
int                                     versionLS = 2;

/* FUNCTION: int main(int argc, char *argv[])
 *
 * PURPOSE: This function is the beginning execution point for the delivery executable.
 *
 * ARGUMENTS:      int          argc          number of command line arguments passed to delivery
 *                char          *argv[]       array of command line argument pointers
 *
 * RETURNS:        None
 *
 * COMMENTS:       None
 *
 */

void main(int argc, char *argv[])
{
    int          iError;

    if ( GetParameters(argc, argv) )
    {
        PrintParameters();
        return;
    }

    if ( (iError=Init()) )
    {
        ErrorMessage(iError);
        Restore();
        return;
    }

    if ( (iError = RunDelivery()) != ERR_SUCCESS )
        ErrorMessage(iError);

    Restore();

    return;
}

/* FUNCTION: void cls(void)
 *
 * PURPOSE: This function clears the console window
 *
 * ARGUMENTS:      None
 *
 * RETURNS:        None
 *
 * COMMENTS:       None
 *
 */

static void cls(void)
{
    HANDLE      hConsole;
    COORD      coordScreen = { 0, 0 }; //here's where we'll home the cursor
    DWORD      cCharsWritten;
    CONSOLE_SCREEN_BUFFER_INFO  csbi; //to get buffer info
    DWORD      dwConSize; //number of character cells in the current buffer

    hConsole = GetStdHandle(STD_OUTPUT_HANDLE);

    //get the number of character cells in the current buffer

    GetConsoleScreenBufferInfo( hConsole, &csbi );
    dwConSize = csbi.dwSize.X * csbi.dwSize.Y;

    //fill the entire screen with blanks
    FillConsoleOutputCharacter( hConsole, (TCHAR) ' ', dwConSize, coordScreen, &cCharsWritten );
    GetConsoleScreenBufferInfo( hConsole, &csbi );

    //now set the buffer's attributes accordingly
    FillConsoleOutputAttribute( hConsole, csbi.wAttributes,dwConSize, coordScreen, &cCharsWritten );

    //put the cursor at (0, 0)
    SetConsoleCursorPosition( hConsole, coordScreen );

    return;
}

/* FUNCTION: int RunDelivery(void)
 *
 * PURPOSE: This function executes the main delivery executable loop.
 *
 * ARGUMENTS:      None
 *
 * RETURNS:        int          ERR_CANNOT_OPEN_PIPE          cannot open named pipe

```

## Appendix A - Application Source Code

---

```
*
*
* ERR_CANNOT_CREATE_THREAD cannot create required threads
* ERR_SUCCESS successfull no
error
*
* COMMENTS: None
*
*/

static int RunDelivery(void)
{
    SECURITY_ATTRIBUTES sa;
    int i;

    cls();

    PrintHeader();

    printf("\n<Starting Delivery Service with %d Threads.>\n", iNumThreads);
    printf("\nPress <Ctrl>C to exit.\n");

    bDone = FALSE;
    _beginthread( CheckKey, 0, NULL );

    printf("\nWaiting for delivery pipe: ");

    while( !bDone )
    {
        AnimateWait1();
        if ( WaitNamedPipe(lpszPipeName, NMPWAIT_USE_DEFAULT_WAIT) )
        {
            sa.nLength = sizeof(sa);
            sa.lpSecurityDescriptor = NULL;
            sa.bInheritHandle = TRUE;

            hPipe = CreateFile(lpszPipeName, GENERIC_READ | GENERIC_WRITE, FILE_SHARE_READ | FILE_SHARE_WRITE, NULL,
OPEN_EXISTING,
FILE_FLAG_OVERLAPPED, NULL);
            if ( hPipe == INVALID_HANDLE_VALUE )
                return ERR_CANNOT_OPEN_PIPE;
            hComPort = CreateIoCompletionPort(hPipe, NULL, 0, 256);
            break;
        }
        Sleep(100);
    }

    if ( !bDone )
    {
        if ( _beginthread( DeliveryHandler, 0, NULL ) == -1 )
            return ERR_CANNOT_CREATE_THREAD;

        for(i=0; i<iNumThreads; i++)
        {
            if ( _beginthread( DeliveryThread, 0, NULL ) == -1 )
                return ERR_CANNOT_CREATE_THREAD;
        }

        printf("\nRunning : ");

        while( !bDone )
            AnimateWait();
    }

    return ERR_SUCCESS;
}

/* FUNCTION: void AnimateWait1(void)
*
* PURPOSE: This function provides a visual indicator that the delivery executable is waiting for
* the delivery pipe to appear.
*
* ARGUMENTS: None
*
* RETURNS: None
*
* COMMENTS: None
*
*/

static void AnimateWait1(void)
{
    const static char szStr[] = "+-|*";
    static char *ptr = (char *)szStr;

    printf("%c\x8", *ptr);
    ptr = (*(ptr+1)) ? ptr + 1 : (char *)szStr;
    Sleep(100);

    return;
}
```



## Appendix A - Application Source Code

---

```
/* FUNCTION: void AnimateWait(void)
 *
 * PURPOSE: This function provides a visual indicator that the delivery executable is waiting for
 *           and processing transactions.
 *
 * ARGUMENTS:      None
 *
 * RETURNS:        None
 *
 * COMMENTS:       None
 *
 */

static void AnimateWait(void)
{
    const static char szStr[] = "%c\\|\\|";
    static char *ptr = (char *)szStr;

    printf("%c\\x8", *ptr);
    ptr = (*(ptr+1)) ? ptr + 1 : (char *)szStr;
    Sleep(100);

    return;
}

/* FUNCTION: int Init(void)
 *
 * PURPOSE: This function prepares the delivery executable for processing.
 *
 * ARGUMENTS:      None
 *
 * RETURNS:        int      iError      Error code if unsuccessful
 *                                     ERR_SUCCESS      No error successful code
 *
 * COMMENTS:       None
 *
 */

static int Init(void)
{
    int      iError;

    InitializeCriticalSection(&WriteLogCriticalSection);
    InitializeCriticalSection(&DeliveryCriticalSection);

    fpLog      = NULL;

    if ( ! (pDeliveryCache = malloc(sizeof(DELIVERY_PACKET) * iQSlots)) )
        return ERR_INSUFFICIENT_MEMORY;

    memset(pDeliveryCache, 0, sizeof(DELIVERY_PACKET) * iQSlots);

    if ( (iError = ReadRegistrySettings()) )
        return iError;

    if ( (iError = OpenLogFile()) )
        return iError;

    //initialize db library for use
    dbinit();

    // install Db Library error and message handlers
    dbmsghandle(DBMSGHANDLE_PROC)msg_handler);
    dberrhandle(DBERRHANDLE_PROC)err_handler);

    return ERR_SUCCESS;
}

/* FUNCTION: void Restore(void)
 *
 * PURPOSE: This function cleans up allocated objects to allow for termination of the
 *           delivery executable.
 *
 * ARGUMENTS:      None
 *
 * RETURNS:        None
 *
 * COMMENTS:       None
 *
 */

static void Restore(void)
{
    int      iret, l, d;

    DeleteCriticalSection(&WriteLogCriticalSection);
    DeleteCriticalSection(&DeliveryCriticalSection);

    l = 1;
}
```

## Appendix A - Application Source Code

---

```
iret = WriteFile(hPipe, &l, 1, &d, NULL);

if ( hPipe != INVALID_HANDLE_VALUE )
    iret = CloseHandle(hPipe);

if ( fpLog )
    fclose(fpLog);

fpLog = NULL;

dbexit();

return;
}

/* FUNCTION: void ErrorMessage(int iError)
 *
 * PURPOSE: This function displays an error message in the delivery executable's console window.
 *
 * ARGUMENTS:      int          iError      error id to be displayed
 *
 * RETURNS:        None
 *
 * COMMENTS:       None
 */
static void ErrorMessage(int iError)
{
    int i;

    static SERRORMSG errorMsgs[] =
    {
        { ERR_SUCCESS, "Success, no error." },
        { ERR_CANNOT_CREATE_THREAD, "Cannot create thread." },
        { ERR_DBGETDATA_FAILED, "Get data failed." },
        { ERR_REGISTRY_NOT_SETUP, "Registry not setup for tpcc." },
        { ERR_CANNOT_ACCESS_DELIVERY_FN, "Cannot access ReadDelivery cache." },
        { ERR_CANNOT_ACCESS_REGISTRY, "Cannot access registry key TPCC." },
        { ERR_CANNOT_CREATE_RESULTS_FILE, "Cannot create results file." },
        { ERR_CANNOT_OPEN_PIPE, "Cannot open delivery pipe." },
        { ERR_READ_PIPE, "Reading Delivery Pipe." },
        { ERR_INSUFFICIENT_MEMORY, "Insufficient memory." },
        { 0, "" }
    };

    for(i=0; errorMsgs[i].szMsg[0]; i++)
    {
        if ( iError == errorMsgs[i].iError )
        {
            printf("\nError(%d): %s", iError, errorMsgs[i].szMsg);
            if ( fpLog )
            {
                EnterCriticalSection(&WriteLogCriticalSection);
                fprintf(fpLog, "**Error(%d): %s\r\n", iError, errorMsgs[i].szMsg);
                if ( bFlush )
                    fflush(fpLog);
                LeaveCriticalSection(&WriteLogCriticalSection);
            }
            return;
        }
    }

    printf("Error(%d): Unknown Error.");
    EnterCriticalSection(&WriteLogCriticalSection);
    fprintf(fpLog, "**Error(%d): Unknown Error.\r\n", iError);
    if ( bFlush )
        fflush(fpLog);
    LeaveCriticalSection(&WriteLogCriticalSection);

    return;
}

/* FUNCTION: BOOL GetParameters(int argc, char *argv[])
 *
 * PURPOSE: This function parses the command line passed in to the delivery executable, initializing
 *          and filling in global variable parameters.
 *
 * ARGUMENTS:      int          argc          number of command line arguments passed to delivery

```

## Appendix A - Application Source Code

---

```
*
* char *argv[] array of command line argument pointers
* RETURNS:      BOOL      FALSE      parameter read successful      TRUE      user has requested parameter information screen be displayed.
*
* COMMENTS:     None
*/

static BOOL GetParameters(int argc, char *argv[])
{
    int i;

    szServer[0] = 0;
    szPassword[0] = 0;
    bFlush = FALSE;
    strcpy(szDatabase, "tpcc");
    strcpy(szUser, "sa");

    for(i=0; i<argc; i++)
    {
        if ( argv[i][0] == '-' || argv[i][0] == '/' )
        {
            switch(argv[i][1])
            {
                case 'S':
                case 's':
                    strcpy(szServer, argv[i]+2);
                    break;

                case 'D':
                case 'd':
                    strcpy(szDatabase, argv[i]+2);
                    break;

                case 'U':
                case 'u':
                    strcpy(szUser, argv[i]+2);
                    break;

                case 'P':
                case 'p':
                    strcpy(szPassword, argv[i]+2);
                    break;

                case 'F':
                case 'f':
                    bFlush = TRUE; //turn on delilog flush when written.
                    break;

                case '?':
                    return TRUE;
            }
        }
    }
    return FALSE;
}

/* FUNCTION: void PrintParameters(void)
*
* PURPOSE: This function displays the supported command line flags.
*
* ARGUMENTS:      None
*
* RETURNS:        None
*
* COMMENTS:       None
*/

static void PrintParameters(void)
{
    PrintHeader();
    printf("DELISRV:\n\n");
    printf("Parameter Default\n");
    printf("-----\n");
    printf("-S Server \n");
    printf("-D Database tpcc \n");
    printf("-U Username sa \n");
    printf("-P Password \n");
    printf("-F Flush output to delilog file when written. OFF \n");
    printf("-? This help screen\n\n");
    printf("Note: Command line switches are NOT case sensitive.\n");

    return;
}

/* FUNCTION: void PrintHeader(void)
*
* PURPOSE: This function displays the delivery executable's banner information.
*
* ARGUMENTS:      None
*
* RETURNS:        None
*/
```

## Appendix A - Application Source Code

---

```
* COMMENTS:      None
*
*/

static void PrintHeader(void)
{
    printf("*****\n");
    printf("**          *\n");
    printf("** Microsoft SQL Server 6.5          *\n");
    printf("**          *\n");
    printf("** HTML TPC-C BENCHMARK KIT: Delivery Server *\n");
    printf("** Version %d.%2d.%3d          *\n", versionMS, versionMM, versionLS);
    printf("**          *\n");
    printf("*****\n");

    return;
}

/* FUNCTION: int ReadRegistrySettings(void)
*
* PURPOSE: This function reads the system registry filling in required key parameters.
*
* ARGUMENTS:      None
*
* RETURNS:        int          ERR_REGISTRY_NOT_SETUP          registry not setup tpcc.exe needs to be run
*
*                to setup registry.
*
*                ERR_SUCCESS          Registry read
*
* Successful, no error
*
*
* COMMENTS:      None
*/

static int ReadRegistrySettings(void)
{
    HKEY    hKey;
    DWORD   size;
    DWORD   type;
    char    szTmp[256];

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

    size = sizeof(szTmp);

    iNumThreads = 4;
    if ( RegQueryValueEx(hKey, "NumberOfDeliveryThreads", 0, &type, szTmp, &size) == ERROR_SUCCESS )
        iNumThreads = atoi(szTmp);
        if ( !iNumThreads )
            iNumThreads = 4;

    iDelayMs = 1000;
    if ( RegQueryValueEx(hKey, "BackoffDelay", 0, &type, szTmp, &size) == ERROR_SUCCESS )
        iDelayMs = atoi(szTmp);
        if ( !iDelayMs )
            iDelayMs = 1000;

    iDeadlockRetry = 3;
    if ( RegQueryValueEx(hKey, "DeadlockRetry", 0, &type, szTmp, &size) == ERROR_SUCCESS )
        iDeadlockRetry = atoi(szTmp);
        if ( !iDeadlockRetry )
            iDeadlockRetry = 3;

    iQSlotts = 3000;
    size = sizeof(szTmp);
    if ( RegQueryValueEx(hKey, "QueueSlotts", 0, &type, szTmp, &size) == ERROR_SUCCESS )
        iQSlotts = atoi(szTmp);
        if ( !iQSlotts )
            iQSlotts = 3000;

    RegCloseKey(hKey);

    return ERR_SUCCESS;
}

/* FUNCTION: void CheckKey(void *ptr)
*
* PURPOSE: This function checks for a key press on the delivery executable's console. If the
*          key press is a Ctrl C then the execution termination flag variable bDone is set to
*          TRUE which will start the termination of the delivery executable.
*
* ARGUMENTS:      void          *ptr          dummy argument passed in though thread manager, unused NULL.
*
* RETURNS:        None
*
* COMMENTS:      None
*/

static void CheckKey(void *ptr)
```

## Appendix A - Application Source Code

---

```
{
    while( !_getch() != CTRL_C)
        ;
    bDone = TRUE;
    return;
}

/* FUNCTION: void DeliveryHandler( void *ptr )
 * PURPOSE: This function is executed in it's own thread what it does is to check for delivery
 * postings in the delivery named pipe. If any are present then it pulls them off ad
 * places them in the next available delivery queue array element.
 *
 * ARGUMENTS:      void      *ptr      dummy argument passed in though thread manager, unused NULL.
 * RETURNS:        None
 * COMMENTS:       None
 */

static void DeliveryHandler( void *ptr )
{
    int      i;
    int      size;
    int      iError;

    while( !bDone )
    {
        for(i=0; i<iQSlots; i++)
        {
            if ( !pDeliveryCache[i].bInUse )
                break;
        }
        if ( i < iQSlots )
        {
            EnterCriticalSection(&DeliveryCriticalSection);
            pDeliveryCache[i].bInUse = TRUE;
            LeaveCriticalSection(&DeliveryCriticalSection);
        }
        else
        {
            EnterCriticalSection(&DeliveryCriticalSection);
            if ( !pDeliveryCache = (LPDELIVERY_PACKET)realloc(pDeliveryCache, sizeof(DELIVERY_PACKET) * (iQSlots+512)))
            {
                ErrorMessage(ERR_INSUFFICIENT_MEMORY);
                LeaveCriticalSection(&DeliveryCriticalSection);
                return;
            }
            for(i=iQSlots; i<iQSlots+512; i++)
                pDeliveryCache[i].bInUse = FALSE;
            i = iQSlots;
            pDeliveryCache[i].bInUse = TRUE;
            LeaveCriticalSection(&DeliveryCriticalSection);
        }

        pDeliveryCache[i].ov.Offset          = i;
        pDeliveryCache[i].ov.Internal        = 0;
        pDeliveryCache[i].ov.InternalHigh   = 0;
        pDeliveryCache[i].ov.OffsetHigh     = 1;
        pDeliveryCache[i].ov.hEvent         = NULL;

        while( !bDone )
        {
            if ( ReadFile(hPipe, &pDeliveryCache[i].trans, sizeof(DELIVERY_TRANSACTION), &size, &pDeliveryCache[i].ov) )
                break;
            if ( bDone )
                break;
            iError = GetLastError();
            if ( iError == ERROR_IO_PENDING )
            {
                while( pDeliveryCache[i].ov.OffsetHigh )
                    Sleep(10);
                break;
            }
            else
            {
                ErrorMessage(ERR_READ_PIPE);
                return;
            }
        }
        Sleep(1);
    }

    return;
}

/* FUNCTION: void DeliveryThread( void *ptr )
 *
 * PURPOSE: This function is executed inside the delivery threads. The queue array
```

## Appendix A - Application Source Code

```
*
*           is continuously check and if any array elements are in use then the
*           array entry is read, cleared and this function processes it.
*
* ARGUMENTS:      void      *ptr      dummy argument passed in though thread manager, unused NULL.
*
* RETURNS:        None
*
* COMMENTS:       The registry key HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TPCC
*                 value NumberOfDeliveryThreads controls how many of these
*                 functions are running. The HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TPCC
*                 value BackoffDelay controls the amount of time this function waits
*                 between checks of the delivery queue.
*
*/

static void DeliveryThread( void *ptr )
{
    int size;
    int key;
    LPOVERLAPPED pov;
    DELIVERY delivery;
    int iError;

    if ( SQLOpenConnection(&delivery.dbproc, szServer, szDatabase, szUser, szPassword, &delivery.spid) )
        return; //error posting tbd

    //while delisrv running i.e. user has not requested termination
    while( !bDone )
    {
        if ( GetQueuedCompletionStatus(hComPort, &size, &key, &pov, (DWORD)-1) )
        {
            pov->OffsetHigh = 0; //clear to notify delivery handler ok to read another entry.
            //some delivery to do so process it
            memcpy(&delivery.queue, &pDeliveryCache[pov->Offset].trans.queue, sizeof(SYSTEMTIME));
            delivery.w_id = pDeliveryCache[pov->Offset].trans.w_id;
            delivery.o_carrier_id = pDeliveryCache[pov->Offset].trans.o_carrier_id;

            if ( (iError=SQLDelivery(&delivery)) )
            {
                ErrorMessage(iError);
                printf("Running : ");
                continue;
            }

            //update log
            WriteLog(&delivery);

            EnterCriticalSection(&DeliveryCriticalSection);
            pDeliveryCache[pov->Offset].bInUse = FALSE;
            LeaveCriticalSection(&DeliveryCriticalSection);
        }
    }

    return;
}

/* FUNCTION: static int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr, char *dberrstr, char *oserrstr)
*
* PURPOSE: This function handles DB-Library errors
*
* ARGUMENTS:      DBPROCESS      *dbproc      DBPROCESS id pointer
*                 int            severity      severity of error
*                 int            dberr        error id
*                 int            oserr        error id
*                 system specific error code
*                 char            *dberrstr    printable error description of dberr
*                 char            *oserrstr    printable error description of oserr
*
* RETURNS:        int            INT_CONTINUE    continue if error is SQLETIME else INT_CANCEL action
*
* COMMENTS:       None
*
*/

static int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr, char *dberrstr, char *oserrstr)
{
    if (oserr != DBNOERR)
        printf("(%d) %s", oserr, oserrstr);

    if ((dbproc == NULL) || (DBDEAD(dbproc)))
        ExitThread((unsigned long)-1);

    return INT_CONTINUE;
}

/* FUNCTION: static 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
*
*/
```

## Appendix A - Application Source Code

---

```
*
*                               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               INT_CANCEL               cancel
operation
*
* COMMENTS:         This function also sets the dead lock dbproc variable if necessary.
*
*/

static int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int severity, char *msgtext)
{
    if ( (msgno == 5701) || (msgno == 2528) || (msgno == 5703) || (msgno == 6006) )
        return INT_CONTINUE;

    // deadlock message
    if (msgno == 1205)
    {
        // set the deadlock indicator
        if (dbgetuserdata(dbproc) != NULL)
            *((BOOL *) dbgetuserdata(dbproc)) = TRUE;
        else
            printf("\nError, dbgetuserdata returned NULL.\n");

        return INT_CONTINUE;
    }

    if (msgno == 0)
        return INT_CONTINUE;
    else
        printf("SQL Server Message (%ld) : %s\n", msgno, msgtext);
    return INT_CANCEL;
}

/* FUNCTION: BOOL SQLOpenConnection(DBPROCESS **dbproc, char *server, char *database, char *user, char *password, int *spid)
*
* PURPOSE: This function opens the sql connection for use.
*
* ARGUMENTS:         DBPROCESS           **dbproc           pointer to returned DBPROCESS
*                               char           *server           SQL server name
*                               char           *database         SQL server database
*                               char           *user             user name
*                               char           *password         user password
*                               int           *spid             pointer to returned spid
*
* RETURNS:           BOOL           FALSE           if successfull
*                               TRUE           if an error occurs
*
* COMMENTS:         None
*
*/

static BOOL SQLOpenConnection(DBPROCESS **dbproc, char *server, char *database, char *user, char *password, int *spid)
{
    LOGINREC *login;

    login = dblogin();
    DBSETLUSER(login, user);
    DBSETLPWD(login, password);

    DBSETLPACKET(login, (USHORT)DEFCLPACKSIZE);

    if ((*dbproc = dbopen(login, server)) == NULL)
        return TRUE;

    // Use the right database
    dbuse(*dbproc, database);

    dbsetuserdata(*dbproc, malloc(sizeof(BOOL)));
    *((BOOL *) dbgetuserdata(*dbproc)) = FALSE;

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

    dbsqlxec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    {
        dbbind(*dbproc, 1, SMALLBIND, (DBINT) 0, (BYTE *) spid);
        while (dbnextrow(*dbproc) != NO_MORE_ROWS);
    }
    dbcmd(*dbproc, "set nocount on");

    dbsqlxec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
        while (dbnextrow(*dbproc) != NO_MORE_ROWS);

    return FALSE;
}
```





# Appendix A - Application Source Code

---

```
static int SQLDelivery(DELIVERY *pDelivery)
{
    RETCODE    rc;
    int        i;
    int        deadlock_count;
    BYTE       *pData;

    deadlock_count = 0;

    // Start new delivery
    while ( TRUE )
    {
        if (dbrpcinit(pDelivery->dbproc, "tpcc_delivery", 0) == SUCCEED)
        {
            dbrpcparam(pDelivery->dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *)&pDelivery->w_id);
            dbrpcparam(pDelivery->dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *)&pDelivery->o_carrier_id);

            if (dbrpcexec(pDelivery->dbproc) == SUCCEED)
            {
                while (((rc = dbresults(pDelivery->dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
                {
                    while (((rc = dbnextrow(pDelivery->dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
                    {
                        for (i=0;i<10;i++)
                        {
                            if(pData=dbdata(pDelivery->dbproc, i+1))
                                pDelivery->o_id[i] = *((DBINT *)pData);
                            else
                                pDelivery->o_id[i] = 0;
                        }
                    }
                }
                if ( !SQLDetectDeadlock(pDelivery->dbproc) )
                    break;
                deadlock_count++;
                Sleep(10 * deadlock_count);
            }
            GetLocalTime(&pDelivery->trans_end);

            return ERR_SUCCESS;
        }
    }

    /* FUNCTION: BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
    *
    * PURPOSE: This function is used to check for deadlock conditions.
    *
    * ARGUMENTS:      DBPROCESS      *dbproc      DBPROCESS to check
    *
    * RETURNS:        BOOL          FALSE          No lock condition present
    *                  TRUE          Lock condition detected
    *
    * COMMENTS:      None
    */

    static BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
    {
        if (*(BOOL *) dbgetuserdata(dbproc) == TRUE)
        {
            *(BOOL *) dbgetuserdata(dbproc) = FALSE;
            return TRUE;
        }
        return FALSE;
    }

    /* FUNCTION: int OpenLogFile(void)
    *
    * PURPOSE: This function opens the delivery log file for use.
    *
    * ARGUMENTS:      None
    *
    * RETURNS:        int          ERR_REGISTRY_NOT_SETUP      Registry not setup.
    *                  ERR_CANNOT_CREATE_RESULTS_FILE        Cannot create results log file.
    *                  ERR_SUCCESS
    *
    *                  Log file successfully opened
    *
    * COMMENTS:      None
    */

    static int OpenLogFile(void)
    {
        HKEY    hKey;
        BOOL    bRc;
        BYTE    szTmp[256];
        char    szKey[256];
        char    szLogPath[256];
    }
}
```

## Appendix A - Application Source Code

---

```
        DWORD    size;
        DWORD    sv;
        int      len;
        char     *ptr;

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

            if ( RegEnumValue(hKey, 0, szKey, &sv, NULL, NULL, szTmp, &size) == ERROR_SUCCESS )
            {
                strcpy(szLogPath, szTmp);
                bRc = FALSE;
            }
            RegCloseKey(hKey);
        }

        if ( bRc )
            return ERR_REGISTRY_NOT_SETUP;

        if ( (ptr = strchr(szLogPath, ','))
            *ptr = 0;

        len = strlen(szLogPath);
        if ( szLogPath[len-1] != '\\')
        {
            szLogPath[len] = '\\';
            szLogPath[len+1] = 0;
        }
        strcat(szLogPath, "delilog.");

        fpLog = fopen(szLogPath, "ab");

        if ( !fpLog )
            return ERR_CANNOT_CREATE_RESULTS_FILE;

        return ERR_SUCCESS;
    }
}
```

### Tpcc.c

```
/*      FILE:                TPCC.C
 *
 *      Microsoft TPC-C Kit Ver. 3.00.000
 *      Audited 08/23/96      By Francois Raab
 *
 *      Copyright Microsoft, 1996
 *
 *      PURPOSE:  Main module for TPCC.DLL which is an ISAPI service dll.
 *      Author:   Philip Durr
 *
 *      philipdu@Microsoft.com
 */

#include <windows.h>
#include <process.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>

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

#include "trans.h" //tpckit transaction header contains definitions of structures specific to
TPC-C
#include "httpext.h" //ISAPI DLL information header

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

char    szServer[32] = { 0 }; //global variables used with this DLL
char    szUser[32]   = { 0 };
char    szPassword[32] = { 0 };
char    szDatabase[32] = "tpcc";
BOOL    bLog        = FALSE;
int     iThreads    = 5;
int     iMaxWareHouses = 500;
int     iDelayMs    = 100;
short   iDeadlockRetry = (short)3;
short   iMaxConnections = (short)25;

//allowable client command strings i.e. CMD=command
```

# Appendix A - Application Source Code

```
char *szCmds[] =
{
    "..NewOrder..", "..Payment..", "..Delivery..", "..Order-Status..", "..Stock-Level..", "..Exit..",
    "Submit", "Begin", "Process", "Menu", "Clear", ""
};

//defined command string functions, called via CMD=command http string from html client.

void (*DoCmd[])(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId) =
{
    NewOrderForm,
    PaymentForm,
    DeliveryForm,
    OrderStatusForm,
    StockLevelForm,
    Exitcmd,
    SubmitCmd,
    BeginCmd,
    ProcessCmd,
    MenuCmd,
    ClearCmd
};

//Terminal client id structure and interface definition
TERM Term = { 0, 0, 0, FALSE, NULL, TermInit, TermAllocate, TermRestore, TermAdd, TermDelete };

//welcome to tpc-c html form buffer, this is first form client sees.
static char *szWelcomeForm = "<HTML>"

C</TITLE></HEAD><BODY>"
session.<BR>"
VALUE="0">"
VALUE="1">"
VALUE="-2">"
VALUE="0">"

VALUE="Submit">"

" <HEAD><TITLE>Welcome To TPC-
"Please Identify your Warehouse and District for this
"<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">"
"<INPUT TYPE=\"hidden\" NAME=\"STATUSID\""
"<INPUT TYPE=\"hidden\" NAME=\"FORMID\""
"<INPUT TYPE=\"hidden\" NAME=\"TERMID\""
"<INPUT TYPE=\"hidden\" NAME=\"SYNCID\""
"Warehouse ID <INPUT NAME=\"w_id\" SIZE=4><BR>"
"District ID <INPUT NAME=\"d_id\" SIZE=2><BR>"
"<HR>"
"<INPUT TYPE=\"submit\" NAME=\"CMD\""
"</FORM><BODY>"
"</HTML>";

static char szTpcLogPath[256]; //path to html log file if logging turned on in registry.
static char szErrorLogPath[256]; //path to error log file.

static CRITICAL_SECTION CriticalSection;
static CRITICAL_SECTION ErrorLogCriticalSection;
static LPTSTR lpszPipeName = TEXT("\\\\.\\pipe\\DELISRV");
static HANDLE hDeliveryWrite = INVALID_HANDLE_VALUE;
static HANDLE hPipe = INVALID_HANDLE_VALUE;

static EXTENSION_CONTROL_BLOCK *gpECB;
static int bTpcExit; //exit delivery disconnect loop as dll exiting.

/* FUNCTION: BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
*
* 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. Connections
* are sent to this function as thread attachments.
*
* ARGUMENTS: HANDLE hModule module handle
* DWORD ul_reason_for_call reason for call
* LPVOID lpReserved reserved for future use
*
* RETURNS: BOOL FALSE errors ocured in initialization
* TRUE DLL successfully initialized
*
* COMMENTS: None
*/

BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID lpReserved)
{
    int i;
    static SECURITY_ATTRIBUTES sa;
    static PSECURITY_DESCRIPTOR pSD;

    switch(ul_reason_for_call)
    {
        case DLL_PROCESS_ATTACH:
            if ( ReadRegistrySettings() )
            {
                MessageBox(NULL, "Cannot Find TPCC Key in registry (run install.exe).", "Init", MB_OK | MB_ICONSTOP);
            }
        }
    }
}
```

## Appendix A - Application Source Code

---

```
        return FALSE;
    }

    InitializeCriticalSection(&CriticalSection);
    InitializeCriticalSection(&ErrorLogCriticalSection);

    (*Term.Init());
    if ( !(*Term.Allocate()) )
    {
        MessageBox(NULL, "Error Trm.Allocate().", "Init", MB_OK | MB_ICONSTOP);
        return FALSE;
    }
    for(i=Term.iNext; i<Term.iAvailable; i++)
        Term.pClientData[i].inUse = 0;
    Term.pClientData[0].inUse = 1;

    // create a security descriptor that allows anyone to access the pipe...
    pSD = (PSECURITY_DESCRIPTOR)malloc( SECURITY_DESCRIPTOR_MIN_LENGTH );
    if ( pSD == NULL )
    {
        MessageBox(NULL, "Error malloc(SEcurity_DESCRIPTOR_MIN_LENGTH)", "Init", MB_OK |
MB_ICONSTOP);
        return FALSE;
    }
    if ( !InitializeSecurityDescriptor(pSD, SECURITY_DESCRIPTOR_REVISION) )
    {
        MessageBox(NULL, "Error InitializeSecurityDescriptor()", "Init", MB_OK | MB_ICONSTOP);
        return FALSE;
    }
    // add a NULL disc. ACL to the security descriptor.
    if ( !SetSecurityDescriptorDacl(pSD, TRUE, (PACL) NULL, FALSE) )
    {
        MessageBox(NULL, "Error SetSecurityDescriptorDacl().", "Init", MB_OK | MB_ICONSTOP);
        return FALSE;
    }

    sa.nLength
sa.lpSecurityDescriptor = pSD;
sa.bInheritHandle = TRUE;
        = sizeof(sa);

    // open delivery named pipe...
    hPipe = CreateNamedPipe(lpszPipeName, FILE_FLAG_OVERLAPPED | PIPE_ACCESS_DUPLEX,
        PIPE_TYPE_BYTE | PIPE_READMODE_BYTE | PIPE_NOWAIT,
        1, 65535, 65535, 250, &sa);

    if ( hPipe == INVALID_HANDLE_VALUE )
    {
        MessageBox(NULL, "Error CreateNamedPipe().", "Init", MB_OK | MB_ICONSTOP);
        free(pSD);
        return FALSE;
    }

    bTpccExit = FALSE;

    if ( _beginthread( DeliveryDixconnect, 0, NULL ) == -1 )
    {
        MessageBox(NULL, "Error _beginthread()", "Init", MB_OK | MB_ICONSTOP);
        return FALSE;
    }

#ifdef USE_ODBC
    if ( SQLAllocEnv(&henv) == SQL_ERROR )
    {
        MessageBox(NULL, "Error SQLAllocEnv()", "Init", MB_OK | MB_ICONSTOP);
        return FALSE;
    }
#else
    dbinit();
    if ( dbgetmaxprocs() < iMaxConnections )
    {
        if ( dbsetmaxprocs(iMaxConnections) == FAIL )
        {
            //set for fail error message when HttpExtensionProc() is called because
            //at this point we don't have a pECB so no way to show error message.
            iMaxConnections = -1;
        }
    }
    // install error and message handlers
    dbmsghandle((DBMSGHANDLE_PROC)msg_handler);
    dberrhandle((DBERRHANDLE_PROC)err_handler);
#endif
    break;
case DLL_THREAD_ATTACH:
    break;
case DLL_THREAD_DETACH:
    break;
case DLL_PROCESS_DETACH:
    if ( pSD )
        free( pSD );

    bTpccExit = TRUE;

```

## Appendix A - Application Source Code

---

```
        if ( hPipe )
            DisconnectNamedPipe(hPipe);

        if (hPipe != INVALID_HANDLE_VALUE)
            CloseHandle(hPipe);

        (*Term.Restore)();

#ifdef USE_ODBC
        SQLFreeEnv(henv);
#else
        dbexit();
#endif

        DeleteCriticalSection(&CriticalSection);
        DeleteCriticalSection(&ErrorLogCriticalSection);

        break;
    }
    return TRUE;
}

/* FUNCTION: void DeliveryDisconnect(void *ptr)
 *
 * PURPOSE: This function handles disconnecting the server side of the delivery pipe when the
 *          delivery handler application shuts down.
 *
 * ARGUMENTS:      void      *ptr      void pointer normally NULL passed from thread handler.
 *
 * RETURNS:        None
 *
 * COMMENTS:       This function runs as thread which allows the client pipe to disconnect by
 *                  sending a byte back though the pipe to the server i.e. this DLL.
 */

static void DeliveryDisconnect(void *ptr)
{
    int                                l, d;
    SECURITY_ATTRIBUTES                 sa;
    PSECURITY_DESCRIPTOR                pSD;

    // create a security descriptor that allows anyone to access the pipe...
    pSD = (PSECURITY_DESCRIPTOR)malloc( SECURITY_DESCRIPTOR_MIN_LENGTH );
    InitializeSecurityDescriptor(pSD, SECURITY_DESCRIPTOR_REVISION);
    SetSecurityDescriptorDacl(pSD, TRUE, (PACL) NULL, FALSE);
    sa.nLength                       = sizeof(sa);
    sa.lpSecurityDescriptor           = pSD;
    sa.bInheritHandle                 = TRUE;

    while( !bTpscExit )
    {
        if ( hPipe && ReadFile(hPipe, &l, 1, &d, NULL ) )
        {
            DisconnectNamedPipe(hPipe);
            CloseHandle(hPipe);
            // open delivery named pipe...
            hPipe = CreateNamedPipe(lpszPipeName, FILE_FLAG_OVERLAPPED | PIPE_ACCESS_DUPLEX,
                                   PIPE_TYPE_BYTE | PIPE_READMODE_BYTE | PIPE_NOWAIT,
                                   1, 65535, 65535, 250, &sa);
        }
        Sleep( 2000 ); //check for delivery application exit once every 2 seconds.
    }

    free(pSD);

    return;
}

/* FUNCTION: BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVer)
 *
 * 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.
 *
 * COMMENTS:       None
 */

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

    return TRUE;
}

/* FUNCTION: DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
```

## Appendix A - Application Source Code

---

```
*
* 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;
    FILE *fp;

    static BOOL bReadRegistry = FALSE;

    if ( iMaxConnections == -1 )
    {
        ErrorMessage(pECB, ERR_CAN_NOT_SET_MAX_CONNECTIONS, ERR_TYPE_WEBDLL, NULL, -1, -1);
        return HSE_STATUS_SUCCESS;
    }

    //if registry setting is for html logging then show http string passed in.
    if ( bLog )
    {
        SYSTEMTIME      systemTime;

        fp = fopen(szTpccLogPath, "ab");

        GetLocalTime(&systemTime);

        fprintf(fp, "* QUERY * %2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
                systemTime.wYear, systemTime.wMonth, systemTime.wDay,
                systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
                pECB->lpszQueryString);

        fclose(fp);
    }

    //process http query
    if ( !ProcessQueryString(pECB, &iCmd, &FormId, &TermId, &iSyncId) )
    {
        if ( TermId < 0 )
            ErrorMessage(pECB, ERR_INVALID_TERMID, ERR_TYPE_WEBDLL, NULL, TermId, iSyncId);
        else
            ErrorMessage(pECB, ERR_COMMAND_UNDEFINED, ERR_TYPE_WEBDLL, NULL, TermId, iSyncId);
        return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
    }

    if ( TermId != 0 )
    {
        if ( !IsValidTermId(TermId) )
        {
            ErrorMessage(pECB, ERR_INVALID_TERMID, ERR_TYPE_WEBDLL, NULL, TermId, iSyncId);
            return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
        }

        //must have a valid syncid here since termid is valid
        if ( iSyncId < 1 || iSyncId != Term.pClientData[TermId].iSyncId )
        {
            ErrorMessage(pECB, ERR_INVALID_SYNC_CONNECTION, ERR_TYPE_WEBDLL, NULL, TermId, iSyncId);
            return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
        }
    }

    //set use time
    Term.pClientData[TermId].iTickCount = GetTickCount();

    //go execute http: command
    (*DoCmd[iCmd])(pECB, FormId, TermId, iSyncId);

    //finish up and keep connection
    return HSE_STATUS_SUCCESS_AND_KEEP_CONN;
}

/* FUNCTION: static BOOL IsValidTermId(int TermId)
*
* PURPOSE: This function checks to see of the passed in terminal id is valid.
*
* ARGUMENTS:      int      TermId      client terminal id
*
* RETURNS:        BOOL      FALSE      Terminal ID Invalid
```

## Appendix A - Application Source Code

---

```
*
* Terminal ID valid
*
* COMMENTS:      None
*
*/

static BOOL IsValidTermId(int TermId)
{
    return (BOOL) ( TermId > 0 && TermId <= Term.iAvailable && Term.pClientData[TermId].inUse );
}

/* FUNCTION: BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB, int *pCmd, int *pFormId, int *pTermId, int *pSyncId)
* PURPOSE: This function extracts the relevant information out of the http command passed in from
*          the browser.
*
* ARGUMENTS:      EXTENSION_CONTROL_BLOCK *pECB          structure pointer to be passed in internet
*                  service information.
*                  returned command id                    int *pCmd
*                  form client browser is on              int *pFormId          returned active
*                  terminal id                            int *pTermId          returned client
*
* RETURNS:        BOOL      FALSE                          success
*                  TRUE
*                  command passed in is invalid
*
* 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.
*/

BOOL ProcessQueryString(EXTENSION_CONTROL_BLOCK *pECB, int *pCmd, int *pFormId, int *pTermId, int *pSyncId)
{
    char *ptr;
    char szBuffer[25];
    char szTmp[25];
    char *dest = szBuffer;
    int i;

    if ( (ptr = strstr(pECB->lpszQueryString, "FORMID=")) )
        *pFormId = *(ptr+7) & 0x0F;

    if ( (ptr = strstr(pECB->lpszQueryString, "TERMINID=")) )
    {
        *pTermId = atoi((ptr+7));
        if ( *pTermId == 0 ) //terminal id 0 used internally
            *pTermId = -1;
        if ( *pTermId == -2 ) //login screen
            *pTermId = 0;
    }
    else
        *pTermId = 0;

    if ( (ptr = strstr(pECB->lpszQueryString, "SYNCID=")) )
        *pSyncId = atoi((ptr+7));
    else
        *pSyncId = 0;

    if ( !(ptr = strstr(pECB->lpszQueryString, "CMD=")) )
    {
        ptr = szBuffer;
        if ( !strcmp(szBuffer, "Default") )
            strcpy(szBuffer, "CMD=Begin");
        switch( *pFormId )
        {
            case WELCOME_FORM:
                strcpy(szBuffer, "CMD=Submit");
                break;
            case MAIN_MENU_FORM:
                strcpy(szBuffer, "CMD=NewOrder");
                break;
            case NEW_ORDER_FORM:
            case PAYMENT_FORM:
            case DELIVERY_FORM:
            case ORDER_STATUS_FORM:
            case STOCK_LEVEL_FORM:
                if ( !(*pTermId) )
                    return FALSE;
                if ( GetKeyValue(pECB->lpszQueryString, "PI*", szTmp, sizeof(szTmp)) )
                    strcpy(szBuffer, "CMD=Process");
                else
                {
                    strcpy(szBuffer, "CMD=");
                    strcat(szBuffer, szCmds[*pFormId - NEW_ORDER_FORM]);
                }
            }
        }
    }
}
```

## Appendix A - Application Source Code

---

```

                                break;
                                return FALSE;
                                }
                                }
                                ptr += 4;
                                while( *ptr && *ptr != '&' )
                                    *dest++ = *ptr++;
                                *dest = 0;
                                for(i=0; szCmds[i][0]; i++)
                                {
                                    if ( !strcmp(szCmds[i], szBuffer) )
                                    {
                                        *pCmd = i;
                                        return TRUE;
                                    }
                                }
                                return FALSE;
}

/* FUNCTION: void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
* PURPOSE: This function wraps the functionality needed for the TPC-C New Order Form.
* ARGUMENTS:      int          int          iFormId          unused
                  id of calling browser, i.e. TERMID= from http command line      iTermId
                  EXTENSION_CONTROL_BLOCK *pECB          structure pointer to passed in internet
                  service information.
* RETURNS:      None
* COMMENTS:      None
*/

void NewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    WriteZString(pECB, MakeNewOrderForm(iTermId, iSyncId, TRUE, FALSE));

    UNUSEDPARAM(iFormId);

    return;
}

/* FUNCTION: void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
* PURPOSE: This function wraps the functionality needed for the TPC-C Payment Form.
* ARGUMENTS:      int          int          iFormId          unused
                  id of calling browser, i.e. TERMID= from http command line      iTermId
                  int          int          iSyncId
                  sync id of calling browser
                  EXTENSION_CONTROL_BLOCK *pECB          structure pointer to passed in internet
                  service information.
* RETURNS:      None
* COMMENTS:      None
*/

void PaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    WriteZString(pECB, MakePaymentForm(iTermId, iSyncId, TRUE));

    UNUSEDPARAM(iFormId);

}

/* FUNCTION: void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
* PURPOSE: This function wraps the functionality needed for the TPC-C Delivery Form.
* ARGUMENTS:      int          int          iFormId          unused
                  id of calling browser, i.e. TERMID= from http command line      iTermId
                  int          int          iSyncId
                  sync id of calling browser
                  EXTENSION_CONTROL_BLOCK *pECB          structure pointer to passed in internet
                  service information.
* RETURNS:      None
* COMMENTS:      None
*/
```



## Appendix A - Application Source Code

---

```
*/
void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    WriteZString(pECB, MakeDeliveryForm(iTermId, iSyncId, TRUE) );

    UNUSEDPARAM(iFormId);
}

/* FUNCTION: void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
* PURPOSE: This function wraps the functionality needed for the TPC-C Order Status Form.
* ARGUMENTS:      int          int          iFormId          unused
                  id of calling browser, i.e. TERMID= from http command line          iTermId
                  sync id of calling browser          int          iSyncId
                  EXTENSION_CONTROL_BLOCK *pECB          structure pointer to passed in internet
                  service information.
* RETURNS:      None
* COMMENTS:      None
*/
void OrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    WriteZString(pECB, MakeOrderStatusForm(iTermId, iSyncId, TRUE) );

    UNUSEDPARAM(iFormId);
}

/* FUNCTION: void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
* PURPOSE: This function wraps the functionality needed for the TPC-C Stock Level Form.
* ARGUMENTS:      int          int          iFormId          unused
                  id of calling browser, i.e. TERMID= from http command line          iTermId
                  sync id of calling browser          int          iSyncId
                  EXTENSION_CONTROL_BLOCK *pECB          structure pointer to passed in internet
                  service information.
* RETURNS:      None
* COMMENTS:      None
*/
void StockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    WriteZString(pECB, MakeStockLevelForm(iTermId, iSyncId, TRUE) );

    return;
}

/* FUNCTION: void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
* PURPOSE: This function removes a terminal id from use, the allocated structure however remains
          valid so the next request for a new client will not require a new memory allocation.
* ARGUMENTS:      int          int          iFormId          unused
                  id of calling browser, i.e. TERMID= from http command line          iTermId
                  sync id of calling browser          int          iSyncId
                  EXTENSION_CONTROL_BLOCK *pECB          structure pointer to passed in internet
                  service information.
* RETURNS:      None
* COMMENTS:      None
*/
void Exitcmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    (*Term.Delete)(pECB, iTermId);

    WriteZString(pECB, MakeWelcomeForm() );

    UNUSEDPARAM(iFormId);
    UNUSEDPARAM(iSyncId);

    return;
}
```

## Appendix A - Application Source Code

---

```
/* FUNCTION: void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE: This function allocated a new terminal id in the Term structure array.
 *
 * ARGUMENTS:      int                                iFormId                unused
 *                  int                                iTermId                iTermId
 *                  id of calling browser, i.e. TERMID= from http command line
 *                  int                                iSyncId                iSyncId
 *                  sync id of calling browser
 *                  EXTENSION_CONTROL_BLOCK *pECB                structure pointer to passed in internet
 *
 *                  service information.
 * RETURNS:      None
 *
 * COMMENTS:      A terminal id can be allocated but still be invalid if the requested warehouse number
 *                  is outside the range specified in the registry. This then will force the client id
 *                  to be invalid and an error message sent to the users browser.
 */
```

```
void SubmitCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    int iCurrent;

    if ( (iCurrent = (*Term.Add)(pECB, pECB->lpszQueryString)) < 0 )
    {
        ErrorMessage(pECB, ERR_CANNOT_INIT_TERMINAL, ERR_TYPE_WEBDLL, NULL, iCurrent, iSyncId);
        return;
    }

    if ( Term.pClientData[iCurrent].w_id > iMaxWareHouses || Term.pClientData[iCurrent].w_id < 1 )
    {
        ErrorMessage(pECB, ERR_W_ID_INVALID, ERR_TYPE_WEBDLL, NULL, iCurrent, iSyncId);
        (*Term.Delete)(pECB, iCurrent);
        return;
    }

    if ( Term.pClientData[iCurrent].d_id < 1 || Term.pClientData[iCurrent].d_id > 10 )
    {
        ErrorMessage(pECB, ERR_D_ID_INVALID, ERR_TYPE_WEBDLL, NULL, iCurrent, iSyncId);
        (*Term.Delete)(pECB, iCurrent);
        return;
    }

    WriteZString(pECB, MakeMainMenuForm(iCurrent, Term.pClientData[iCurrent], iSyncId));

    return;
}
```

```
/* FUNCTION: void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE: This function is the first command executed. It is executed with the command
 *          CMD=Begin?Server=xxx from the http command line.
 *
 * ARGUMENTS:      int                                iFormId                unused
 *                  int                                iTermId                iTermId
 *                  id of calling browser, i.e. TERMID= from http command line
 *                  int                                iSyncId                iSyncId
 *                  sync id of calling browser
 *                  EXTENSION_CONTROL_BLOCK *pECB                structure pointer to passed in internet
 *
 *                  service information.
 * RETURNS:      None
 *
 * COMMENTS:      SQL server must be specified, however the user and password parameters are optional.
 *                  The complete command line is CMD=Begin&Server=server&User=sa&Psw=&. The & are used
 *                  to separate parameters which is internet browser standard.
 */
```

```
void BeginCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    LPSTR pQueryString;

    pQueryString = pECB->lpszQueryString;

    if ( !GetKeyValue(pQueryString, "Server", szServer, sizeof(szServer)) )
    {
        ErrorMessage(pECB, ERR_NO_SERVER_SPECIFIED, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    if ( !GetKeyValue(pQueryString, "User", szUser, sizeof(szUser)) )
        strcpy(szUser, "sa");

    if ( !GetKeyValue(pQueryString, "Psw", szPassword, sizeof(szPassword)) )
        strcpy(szPassword, "");

    if ( !GetKeyValue(pQueryString, "Db", szDatabase, sizeof(szDatabase)) )
        strcpy(szDatabase, "tpcc");

    WriteZString(pECB, MakeWelcomeForm());

    UNUSEDPARAM(iFormId);
}
```

## Appendix A - Application Source Code

---

```
        return;
    }

/* FUNCTION: void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE: This function process the passed in http command
 *
 * ARGUMENTS:          int          iFormId          unused
 *                   int          iTermId         iTermId
 *                   id of calling browser, i.e. TERMID= from http command line
 *                   int          iSyncId         iSyncId
 *                   sync id of calling browser
 *                   EXTENSION_CONTROL_BLOCK *pECB          structure pointer to passed in internet
 *                   service information.
 * RETURNS:           None
 *
 * COMMENTS:         None
 *
 */

void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    switch( iFormId )
    {
        case WELCOME_FORM:
            return;
        case MAIN_MENU_FORM:
            return;
        case NEW_ORDER_FORM:
            ProcessNewOrderForm(pECB, iTermId, iSyncId);
            return;
        case PAYMENT_FORM:
            ProcessPaymentForm(pECB, iTermId, iSyncId);
            return;
        case DELIVERY_FORM:
            ProcessDeliveryForm(pECB, iTermId, iSyncId);
            return;
        case ORDER_STATUS_FORM:
            ProcessOrderStatusForm(pECB, iTermId, iSyncId);
            return;
        case STOCK_LEVEL_FORM:
            ProcessStockLevelForm(pECB, iTermId, iSyncId);
            return;
    }
}

/* FUNCTION: void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE: This function frees all currently logged in terminal ids.
 *
 * ARGUMENTS:          int          iFormId          unused
 *                   int          iTermId         iTermId
 *                   id of calling browser, i.e. TERMID= from http command line
 *                   int          iSyncId         iSyncId
 *                   sync id of calling browser
 *                   EXTENSION_CONTROL_BLOCK *pECB          structure pointer to passed in internet
 *                   service information.
 * RETURNS:           None
 *
 * COMMENTS:         Use this function with caution, it may cause unpredictable results
 *                   if existing browsers attempt to use the web client with out
 *                   beginning at the login screen for each client.
 *
 */

void ClearCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    int i;

    EnterCriticalSection(&CriticalSection);

    for(i=0; i<Term.iAvailable; i++)
    {
        if ( Term.pClientData[i].inUse )
            (*Term.Delete)(pECB, i);
    }

    Term.iNext          = 0;
    Term.iAvailable     = 0;
    Term.iMasterSyncId = 1;

    if ( Term.pClientData )
        free(Term.pClientData);
    Term.pClientData    = NULL;
    Term.bInit          = FALSE;

    (*Term.Init());
    if ( !(*Term.Allocate()) )
    {

```

## Appendix A - Application Source Code

---

```
        ErrorMessage(pECB, ERR_MAX_CONNECT_PARAM, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }
    for(i=Term.iNext; i<Term.iAvailable; i++)
        Term.pClientData[i].inUse = 0;
    Term.pClientData[0].inUse = 1;

    LeaveCriticalSection(&CriticalSection);

    WriteZString(pECB, MakeWelcomeForm() );

    return;
}

/* FUNCTION: void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
 *
 * PURPOSE: This function causes an exit to the main menu
 *
 * ARGUMENTS:      int          iFormId          unused
 *                int          iTermId         unused
 *                id of calling browser, i.e. TERMID= from http command line
 *                int          iSyncId         unused
 *                sync id of calling browser
 *                EXTENSION_CONTROL_BLOCK *pECB      structure pointer to passed in internet
 *                service information.
 * RETURNS:      None
 *
 * COMMENTS:      None
 */
void MenuCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iTermId, int iSyncId)
{
    WriteZString(pECB, MakeMainMenuForm(iTermId, iSyncId) );

    return;
}

/* FUNCTION: void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr)
 *
 * PURPOSE: This function is the low level output function. It writes a string of text back to the
 *          client browser.
 *
 * ARGUMENTS:      EXTENSION_CONTROL_BLOCK *pECB      passed in structure pointer from inetrv.
 *                char          *szStr                string to display in the client
 *                browser.
 * RETURNS:      None
 *
 * COMMENTS:      This function assumes that the string to written to the client browser has
 *                been formatted in an HTML manner.
 */
static void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr)
{
    FILE *fp;
    int lpbSize;
    int iSize;
    char szHeader[128];
    char szHeader1[128];

    lpbSize = strlen(szStr)+1;

    if ( bLog )
    {
        SYSTEMTIME      systemTime;

        fp = fopen(szTpccLogPath, "ab");

        GetLocalTime(&systemTime);

        fprintf(fp, "** HTML PAGE * %2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\ns\r\n\r\n",
                systemTime.wYear, systemTime.wMonth, systemTime.wDay,
                systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
                szStr);

        fclose(fp);
    }

    iSize = sprintf(szHeader, "200 Ok");
    sprintf(szHeader1, "Connection: keep-alive\r\nContent-type: text/html\r\nContent-length: %d\r\n\r\n", lpbSize);

    (*pECB->ServerSupportFunction)(pECB->ConnID, HSE_REQ_SEND_RESPONSE_HEADER, szHeader, &iSize, (LPDWORD)szHeader1);
    (*pECB->WriteClient)(pECB->ConnID, szStr, &lpbSize, 0);

    return;
}
}
```

## Appendix A - Application Source Code

---

```
/* FUNCTION: void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format, ...)
 *
 * PURPOSE: This function forms a high level printf for an HTML browser
 *
 * ARGUMENTS:      EXTENSION_CONTROL_BLOCK *pECB    passed in structure pointer from inetsrv.
 *                  char *format                    printf style format string
 *                  ...
 *                  other arguments as required by printf style format string.
 *
 * RETURNS:        None
 *
 * COMMENTS:       This function is mainly used for developmental support.
 */

static void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format, ...)
{
    int lpbSize;
    char szBuff[512];
    char szTmp[512];

    va_list marker;
    va_start( marker, format );
    vsprintf(szTmp, format, marker);
    va_end( marker );

    lpbSize = wsprintf(szBuff, "<html>%s</html>", szTmp) + 1;

    (*pECB->WriteClient)(pECB->ConnID, szBuff, &lpbSize, 0);

    return;
}

/* FUNCTION: void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int iErrorType, char *szMsg)
 *
 * PURPOSE: This function displays an error message in the client browser.
 *
 * ARGUMENTS:      EXTENSION_CONTROL_BLOCK *pECB    passed in structure pointer from inetsrv.
 *                  int iError                       id of error
 *                  int iErrorType                   error type,
 *                  ERR_TYPE_SQL, ERR_TYPE_DBLIB, or ERR_TYPE_WEBDLL
 *                  int iTermId                      terminal id
 *                  int iSyncid                      sync id from
 *                  char *szMsg                      optional error message string
 *                  used with ERR_TYPE_SQL and
 *                  ERR_TYPE_DBLIB
 *
 * RETURNS:        None
 *
 * COMMENTS:       If the error type is ERR_TYPE_WEBDLL the szmsg parameter may be NULL because it
 *                  is ignored. If the error type is ERR_TYPE_SQL or ERR_TYPE_DBLIB then the szMsg
 *                  parameter contains the text of the error message, so the szMsg parameter cannot
 *                  be NULL.
 */

void ErrorMessage(EXTENSION_CONTROL_BLOCK *pECB, int iError, int iErrorType, char *szMsg, int iTermId, int iSyncId)
{
    int i;

    static SERRORMSG errorMsgs[] =
    {
        { ERR_SUCCESS, "Success, no error." },
        { ERR_COMMAND_UNDEFINED, "Command undefined." },
        { ERR_NOT_IMPLEMENTED_YET, "Not Implemented Yet." },
        { ERR_CANNOT_INIT_TERMINAL, "Cannot initialize client connection." },
        { ERR_OUT_OF_MEMORY, "insufficient memory." },
        { ERR_NEW_ORDER_NOT_PROCESSED, "Cannot process new Order form." },
        { ERR_PAYMENT_NOT_PROCESSED, "Cannot process payment form." },
        { ERR_NO_SERVER_SPECIFIED, "No Server name specified." },
        { ERR_ORDER_STATUS_NOT_PROCESSED, "Cannot process order status form." },
        { ERR_W_ID_INVALID, "Invalid Warehouse ID." },
        { ERR_CAN_NOT_SET_MAX_CONNECTIONS, "Insufficient memory to allocate # connections." },
        { ERR_NOSUCH_CUSTOMER, "No such customer." }
    },
    i;
}
```

## Appendix A - Application Source Code

```

to 10."          {      ERR_D_ID_INVALID,                                },                "Invalid District ID Must be 1
exceeded, run install to increase." {      ERR_MAX_CONNECT_PARAM,                                },                "Max client connections
{      ERR_INVALID_SYNC_CONNECTION,                        },                "Invalid Terminal Sync ID."
{      ERR_INVALID_TERMID,                                },                "Invalid Terminal ID."
{      ERR_PAYMENT_INVALID_CUSTOMER,                      },                "Payment Form, No such Customer."
{      ERR_SQL_OPEN_CONNECTION,                          },                "SQLOpenConnection API Failed."
{      ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY,              },                "Stock Level missing Threshold key \"TT*\"."
{      ERR_STOCKLEVEL_THRESHOLD_INVALID,                  },                "Stock Level Threshold invalid data type range = 1 - 99."
},
{      ERR_STOCKLEVEL_THRESHOLD_RANGE,                    },                "Stock Level Threshold out of range, range
must be 1 - 99." {      ERR_STOCKLEVEL_NOT_PROCESSED,                      },                "Stock Level not processed."
{      ERR_NEWORDER_FORM_MISSING_DID,                     },                "New Order missing District key \"DID*\"."
10."            {      ERR_NEWORDER_DISTRICT_INVALID,                   },                "New Order District ID Invalid range 1 -
{      ERR_NEWORDER_DISTRICT_RANGE,                       },                "New Order District ID out of Range. Range = 1 - 10."
{      ERR_NEWORDER_CUSTOMER_KEY,                         },                "New Order missing Customer key
\"CID*\"."      {      ERR_NEWORDER_CUSTOMER_INVALID,                       },                "New Order customer id invalid data type,
range = 1 to 3000." {      ERR_NEWORDER_CUSTOMER_RANGE,                       },                "New Order customer id out of range, range
= 1 to 3000." {      ERR_NEWORDER_MISSING_IID_KEY,                       },                "New Order missing Item Id key \"IID*\"."
{      ERR_NEWORDER_ITEM_BLANK_LINES,                     },                "New Order blank order lines all orders must
be continuous." {      ERR_NEWORDER_ITEMID_INVALID,                       },                "New Order Item Id is wrong data type, must be numeric."
{      ERR_NEWORDER_MISSING_SUPPW_KEY,                    },                "New Order missing Supp_W key
\"SP##*\"."    {      ERR_NEWORDER_SUPPW_INVALID,                       },                "New Order Supp_W invalid data type must
be numeric." {      ERR_NEWORDER_MISSING_QTY_KEY,                      },                "New Order Missing Qty key \"Qty##*\"."
{      ERR_NEWORDER_QTY_INVALID,                          },                "New Order Qty invalid must be numeric
range 1 - 99." {      ERR_NEWORDER_SUPPW_RANGE,                          },                "New Order Supp_W value out of range
range = 1 - Max Warehouses." {      ERR_NEWORDER_ITEMID_RANGE,                        },                "New Order Item Id is out of range. Range =
1 to 999999." {      ERR_NEWORDER_QTY_RANGE,                            },                "New Order Qty is out of
range. Range = 1 to 99." {      ERR_PAYMENT_DISTRICT_INVALID,                    },                "Payment District ID is invalid must be 1 - 10."
Item_Id."      }, {      ERR_NEWORDER_SUPPW_WITHOUT_ITEMID,                },                "New Order Supp_W field entered without a corresponding
{      ERR_NEWORDER_QTY_WITHOUT_ITEMID,                   },                "New Order Qty entered without a corresponding Item_Id."
},
{      ERR_NEWORDER_NOITEMS_ENTERED,                     },                "New Order Blank Items between items,
items must be continuous." {      ERR_PAYMENT_MISSING_DID_KEY,                      },                "Payment missing District Key \"DID*\"."
{      ERR_PAYMENT_DISTRICT_RANGE,                       },                "Payment District Out of range, range = 1 -
10." {      ERR_PAYMENT_MISSING_CID_KEY,                      },                "Payment missing Customer Key \"CID*\"."
{      ERR_PAYMENT_CUSTOMER_INVALID,                     },                "Payment Customer data type invalid, must
be numeric." {      ERR_PAYMENT_MISSING_CLT,                       },                "Payment missing Customer Last Name Key
\"CLT*\"."    {      ERR_PAYMENT_LAST_NAME_TO_LONG,                    },                "Payment Customer last name longer than 16
characters." {      ERR_PAYMENT_CUSTOMER_RANGE,                    },                "Payment Customer ID out of range, must
be 1 to 3000." {      ERR_PAYMENT_CID_AND_CLT,                          },                "Payment Customer ID and Last Name
entered must be one or other." {      ERR_PAYMENT_MISSING_CDI_KEY,                      },                "Payment missing Customer district key \"CDI*\"."
{      ERR_PAYMENT_CDI_INVALID,                          },                "Payment Customer district invalid must be
numeric." {      ERR_PAYMENT_CDI_RANGE,                          },                "Payment Customer district
out of range must be 1 - 10." {      ERR_PAYMENT_MISSING_CWI_KEY,                      },                "Payment missing Customer Warehouse key \"CWI*\"."
{      ERR_PAYMENT_CWI_INVALID,                          },                "Payment Customer Warehouse invalid must
be numeric." {      ERR_PAYMENT_CWI_RANGE,                          },                "Payment Customer
Warehouse out of range, 1 to Max Warehouses." {      ERR_PAYMENT_MISSING_HAM_KEY,                      },                "Payment missing Amount key \"HAM*\"."
},

```

## Appendix A - Application Source Code

```

numeric."      {      ERR_PAYMENT_HAM_INVALID,      "Payment Amount invalid data type must be
                },
range, 0 - 9999.99." {      ERR_PAYMENT_HAM_RANGE,      "Payment Amount out of
                },
                ERR_ORDERSTATUS_MISSING_DID_KEY,      "Order Status missing District key \"DID*\"."
                },
10."          {      ERR_ORDERSTATUS_DID_INVALID,      "Order Status District invalid, value mustbe numeric 1 -
                },
- 10."        {      ERR_ORDERSTATUS_DID_RANGE,      "Order Status District out of range must be 1
                },
                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_CLT_RANGE,      "Order Status Customer last name longer
                },
than 16 characters." {      ERR_ORDERSTATUS_CID_INVALID,      "Order Status Customer ID invalid, range must be numeric
                },
1 - 3000."    {      ERR_ORDERSTATUS_CID_RANGE,      "Order Status Customer ID out of range
                },
must be 1 - 3000." {      ERR_ORDERSTATUS_CID_AND_CLT,      "Order Status Customer ID and LastName entered must be
                },
only one."    {      ERR_DELIVERY_MISSING_OCD_KEY,      "Delivery missing Carrier ID key \"OCD*\"."
                },
                ERR_DELIVERY_CARRIER_INVALID,      "Delivery Carrier ID invalid must be numeric 1 - 10."
                },
                ERR_DELIVERY_CARRIER_ID_RANGE,      "Delivery Carrier ID out of range must be 1 -
                },
10."         {      ERR_PAYMENT_MISSING_CLT_KEY,      "Payment missing Customer Last Name key \"CLT*\"."
                },
                },
...          {      0,
                },
                }
};

static char szNoMsg[] = "";
char      *szForm;

if ( !szMsg )
    szMsg = szNoMsg;

if ( iTermId > 0 && IsValidTermId(iTermId) )
    szForm = Term.pClientData[iTermId].szBuffer; //if termid valid use common terminal static buffer.
else
    szForm = Term.pClientData[0].szBuffer; //else term id invalid so use common terminal static buffer.
switch(iErrorType)
{
    case ERR_TYPE_WEBDLL:
        for(i=0; errorMsgs[i].szMsg[0]; i++)
        {
            if ( iError == errorMsgs[i].iError )
                break;
        }
        if ( !errorMsgs[i].szMsg[0] )
            i = 1;
        strcpy(szForm, " <HTML><HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY><FORM
ACTION=\"tpcc.dll\" METHOD=\"GET\">");
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iError);
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
        sprintf(szForm+strlen(szForm), "Error: TPCCWEB(%d): %s", iError, errorMsgs[i].szMsg);
        strcat(szForm, "</FORM><BODY></HTML>");
        WriteZString(pECB, szForm);
        break;
    case ERR_TYPE_SQL:
        strcpy(szForm, " <HTML><HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY><FORM
ACTION=\"tpcc.dll\" METHOD=\"GET\">");
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iError);
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
        sprintf(szForm+strlen(szForm), "Error: SQLSVR(%d): %s", iError, szMsg);
        strcat(szForm, "</FORM><BODY></HTML>");
        WriteZString(pECB, szForm);
        break;
    case ERR_TYPE_DBLIB:
        strcpy(szForm, " <HTML><HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY><FORM
ACTION=\"tpcc.dll\" METHOD=\"GET\">");
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"%d\">", iError);
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"TERMID\" VALUE=\"%d\">", iTermId);
        sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
        sprintf(szForm+strlen(szForm), "Error: DBLIB(%d): %s", iError, szMsg);
        strcat(szForm, "</FORM><BODY></HTML>");
        WriteZString(pECB, szForm);
        break;
}
return;
}

/* FUNCTION: BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue, int iMax)
 *
 * PURPOSE: This function parses a http formatted string for specific key values.

```

## Appendix A - Application Source Code

---

```
*
* 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.
*
* RETURNS:        BOOL         FALSE      key value not found
*                TRUE         TRUE       key valud found
*
*
* COMMENTS:       http keys are formatted either KEY=value& or KEY=value0. This DLL formats
*                TPC-C input fields in such a manner that the keys can be extracted in the
*                above manner.
*/

static BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue, int iMax)
{
    char *ptr;

    if ( !(ptr=strstr(pQueryString, pKey)) )
        return FALSE;
    if ( !(ptr=strchr(ptr, '=')) )
        return FALSE;
    ptr++;
    iMax--;
    while( *ptr && *ptr != '&' && iMax)
    {
        *pValue++ = *ptr++;
        iMax--;
    }
    *pValue = 0;
    return TRUE;
}

/* FUNCTION: void Termlnit(void)
*
* PURPOSE: This function initializes the client terminal structure it is calld when the TPCC.DLL
*          is first loaded by the inet service.
*
* ARGUMENTS:      none
*
* RETURNS:        None
*
* COMMENTS:       None
*/

static void Termlnit(void)
{
    if ( Term.bInIt )
        return;
    Term.iNext          = 0;
    Term.iMasterSyncId = 1;
    Term.iAvailable     = 0;
    Term.pClientData    = NULL;
    Term.bInIt         = TRUE;
    return;
}

#ifndef USE_ODBC
/* FUNCTION: int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr, char *dberrstr, char *oserrstr)
*
* PURPOSE: This function handles DB-Library errors
*
* ARGUMENTS:      DBPROCESS *dbproc          DBPROCESS id pointer
*                int          severity       severity of
error
*                int          dberr         dberr
*                int          oserr        oserr
*                char          *dberrstr    printable error description of
dberr
*                char          *oserrstr    printable error description of
oserr
*
* RETURNS:        int          INT_CONTINUE  continue if error is SQLETIME else
INT_CANCEL action
*
* COMMENTS:       None
*/

int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr, char *dberrstr, char *oserrstr)
{

```



# Appendix A - Application Source Code

```

PECBINFO
EXTENSION_CONTROL_BLOCK *pECB; pEcbInfo;
FILE *fp;
SYSTEMTIME systemTime;
char szTmp[256];
int iTermId;
int iSyncId;

pEcbInfo = NULL;

if ((dbproc == NULL) || (DBDEAD(dbproc)))
{
    ErrorMessage(gpECB, -1, ERR_TYPE_DBLIB, "DBPROC is invalid.", iTermId, iSyncId);
    return INT_CANCEL;
}

if (!pEcbInfo = (PECBINFO)dbgetuserdata(dbproc))
{
    pECB = gpECB;
    iTermId = 0;
    iSyncId = 0;
}
else
{
    pECB = pEcbInfo->pECB;
    iTermId = pEcbInfo->iTermId;
    iSyncId = pEcbInfo->iSyncId;
}

if (pEcbInfo && pEcbInfo->bFailed)
    return INT_CANCEL;

if (oserr != DBNOERR)
{
    ErrorMessage(pECB, oserr, ERR_TYPE_DBLIB, oserrstr, iTermId, iSyncId);

    if (pEcbInfo)
        pEcbInfo->bFailed = TRUE;

    GetLocalTime(&systemTime);
    fp = fopen(szErrorLogPath, "ab");

    EnterCriticalSection(&ErrorLogCriticalSection);

    sprintf(szTmp, "Error: DBLIB(%d): %s", oserr, oserrstr);

    fprintf(fp, "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
            systemTime.wYear, systemTime.wMonth, systemTime.wDay,
            systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
            szTmp);
    LeaveCriticalSection(&ErrorLogCriticalSection);

    fclose(fp);
}

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
 *            int msgstate message state
 *            int severity message
 *            char *msgtext printable message description
 *
 * RETURNS: int INT_CONTINUE continue if error is SQLETIME else
 *          INT_CANCEL cancel operation
 *
 * COMMENTS: This function also sets the dead lock dbproc variable if necessary.
 */

int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int severity, char *msgtext)
{
    PECBINFO
    EXTENSION_CONTROL_BLOCK *pECB; pEcbInfo;
    FILE *fp;
    SYSTEMTIME systemTime;
    char szTmp[256];
    int iTermId;
    int iSyncId;

```

## Appendix A - Application Source Code

---

```
if ( !(pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
{
    pECB = gpECB;
    iTermId = 0;
    iSyncId = 0;
}
else
{
    pECB = pEcbInfo->pECB;
    iTermId = pEcbInfo->iTermId;
    iSyncId = pEcbInfo->iSyncId;
}

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

// deadlock message
if (msgno == 1205)
{
    // set the deadlock indicator
    if ( pEcbInfo )
        pEcbInfo->bDeadlock = TRUE;
    else
        ErrorMessage(pECB, -1, ERR_TYPE_SQL, "Error, dbgetuserdata returned NULL.", iTermId, iSyncId);
    return INT_CONTINUE;
}
if ( pEcbInfo && pEcbInfo->bFailed )
    return INT_CANCEL;

if (msgno == 0)
    return INT_CONTINUE;
else
{
    ErrorMessage(pECB, msgno, ERR_TYPE_SQL, msgtext, iTermId, iSyncId);

    if ( pEcbInfo )
        pEcbInfo->bFailed = TRUE;

    GetLocalTime(&systemTime);
    fp = fopen(szErrorLogPath, "ab");

    EnterCriticalSection(&ErrorLogCriticalSection);
    sprintf(szTmp, "Error: SQLSVR(%d): %s", msgno, msgtext);
    fprintf(fp, "%2.2d/%2.2d/%2.2d %2.2d:%2.2d:%2.2d\r\n\r\n%s\r\n\r\n",
            systemTime.wYear, systemTime.wMonth, systemTime.wDay,
            systemTime.wHour, systemTime.wMinute, systemTime.wSecond,
            szTmp);
    LeaveCriticalSection(&ErrorLogCriticalSection);

    fclose(fp);
}

return INT_CANCEL;
}

#endif

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

static void TermRestore(void)
{
    Term.iNext          = 0;
    Term.iAvailable     = 0;
    Term.iMasterSyncId = 0;
    if ( Term.pClientData )
        free(Term.pClientData);
    Term.pClientData    = NULL;
    Term.bInit          = FALSE;

    return;
}

/* FUNCTION: int TermAllocate(void)
*
* PURPOSE: This function allocates more terminal array entries in the Term structure.
*
* ARGUMENTS: None
*
* RETURNS:    int      TRUE or 1 if successfull
*             int      FALSE or 0 if terminal id cannot be allocated.
```

## Appendix A - Application Source Code

---

```
*
* COMMENTS:      None
*
*/

static int TermAllocate(void)
{
    Term.iAvailable += 32;
    if ( !Term.pClientData )
        Term.pClientData = (PCLIENTDATA)malloc(Term.iAvailable * sizeof(CLIENTDATA));
    else
        Term.pClientData = (PCLIENTDATA)realloc(Term.pClientData, Term.iAvailable * sizeof(CLIENTDATA));
    return ( Term.pClientData ) ? 1 : 0;
}

/* FUNCTION: int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString)
*
* PURPOSE: This function assigns a terminal id which is used to identify a client browser.
*
* ARGUMENTS:      EXTENSION_CONTROL_BLOCK      *pECB                passed in structure pointer from inetsrv.
                  char                          *pQueryStringhttp query string passed to
this DLL.
*
* RETURNS:       int                          assigned terminal id
                  -1                          cannot assign id error occurred.
*
* COMMENTS:      if the terminal id cannot be assigned it is because of insufficient memory or the
                  SQL connection cannot be allocated.
*/

static int TermAdd(EXTENSION_CONTROL_BLOCK *pECB, char *pQueryString)
{
    char          szTmp[32];
    int           i, iCurrent, iTotConnections, iTickCount;

    EnterCriticalSection(&CriticalSection);

    for(i=0, iTotConnections = 0; i<Term.iAvailable; i++)
    {
        if ( Term.pClientData[i].inUse )
            iTotConnections++;
    }

    if ( iTotConnections >= iMaxConnections )
    {
        for(iCurrent = 1, i=1, iTickCount = 0x7FFFFFFF; i<iMaxConnections; i++)
        {
            if ( iTickCount > Term.pClientData[i].iTickCount )
            {
                iTickCount = Term.pClientData[i].iTickCount;
                iCurrent = i;
            }
        }
    }
    else
    {
        for(i=0; i<Term.iAvailable; i++)
        {
            if ( !Term.pClientData[i].inUse )
                break;
        }
        iCurrent = i;
    }

    if ( i == Term.iAvailable )
    {
        Term.iNext = Term.iAvailable;
        if ( !(*Term.Allocate)() )
            goto TermAddErr1;
        for(i=Term.iNext; i<Term.iAvailable; i++)
            Term.pClientData[i].inUse = 0;
        iCurrent = Term.iNext;
    }

    Term.pClientData[iCurrent].inUse = 1;

    if ( !GetKeyValue(pQueryString, "w_id", szTmp, sizeof(szTmp)) )
        goto TermAddErr1;

    Term.pClientData[iCurrent].w_id = (short)atoi(szTmp);

    if ( !GetKeyValue(pQueryString, "d_id", szTmp, sizeof(szTmp)) )
        goto TermAddErr1;

    Term.pClientData[iCurrent].d_id = atoi(szTmp);

    Term.pClientData[iCurrent].iTickCount = GetTickCount();
    Term.pClientData[iCurrent].iSynclD = Term.iMasterSynclD++;
}
```

## Appendix A - Application Source Code

---

```
if ( Init(pECB, iCurrent, Term.pClientData[iCurrent], iSyncId, szServer, szUser, szPassword, szDatabase) )
{
    (*Term.Delete)(pECB, iCurrent);
    goto TermAddErr1;
}

LeaveCriticalSection(&CriticalSection);
return iCurrent;

TermAddErr1:
LeaveCriticalSection(&CriticalSection);
return -1; //terminal unsuccessfully added
}

/* FUNCTION: void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int id)
*
* PURPOSE: This function makes a terminal entry in the Term array available for reuse.
*
* ARGUMENTS:      EXTENSION_CONTROL_BLOCK    *pECB    passed in structure pointer from inetrv.
*                  int                        id
*                  Terminal id of client exiting
*
* RETURNS:        None
*
* COMMENTS:       None
*/

static void TermDelete(EXTENSION_CONTROL_BLOCK *pECB, int id)
{
    if ( id >= 0 && id < Term.iAvailable )
    {
        Close(pECB, id, -1);
        Term.pClientData[id].inUse = 0;
    }

    return;
}

/* FUNCTION: BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, char *szServer, char *szUser, char *szPassword, char *szDatabase)
*
* PURPOSE: This function initializes the sql connection for use.
*
* ARGUMENTS:      EXTENSION_CONTROL_BLOCK    *pECB    passed in structure pointer from inetrv.
*                  int                        iTermId
*                  id of browser client that this connection is for.
*                  int                        iSyncId
*                  sync id for this client session
*                  char                        *szServer    sql server name
*                  char                        *szUser      user name
*                  char                        *szPassword  user password
*                  char                        *szDatabase  database to use
*
* RETURNS:        BOOL    FALSE    if successfull
*                  TRUE    if an error occurs and connection cannot be established.
*
* COMMENTS:       None
*/

BOOL Init(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, char *szServer, char *szUser, char *szPassword, char *szDatabase)
{
    char    szApp[32];

    sprintf(szApp, "TPCC:%ld", (int)iTermId);

    Term.pClientData[iTermId].dbproc = NULL;

    if ( SQLOpenConnection(pECB, iTermId, iSyncId, &Term.pClientData[iTermId].dbproc, szServer, szDatabase, szUser, szPassword, szApp,
&Term.pClientData[iTermId].spid) )
    {
        ErrorMessage(pECB, ERR_SQL_OPEN_CONNECTION, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return TRUE;
    }

    return FALSE;
}

/* FUNCTION: BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
*
* PURPOSE: This function closes the sql connection for use.
*
* ARGUMENTS:      EXTENSION_CONTROL_BLOCK    *pECB    passed in structure pointer from inetrv.
*                  int                        iTermId    id of browser
*                  int                        iSyncId    sync id of
*                  client browser
*
* RETURNS:        BOOL    FALSE    if successfull
*                  TRUE    if an error occurs and connection cannot be terminated.
*
* COMMENTS:       None
*/
```

## Appendix A - Application Source Code

---

```
*
*/
static BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
{
    PECBINFO pEcbInfo;

    if (Term.pClientData[iTermId].dbproc != NULL)
    {
        if ( (pEcbInfo = (PECBINFO)dbgetuserdata(Term.pClientData[iTermId].dbproc)) )
        {
            pEcbInfo->iTermId = -1;
            pEcbInfo->iSyncId = -1;
            free(pEcbInfo); //free up user info
        }
        return SQLCloseConnection(pECB, Term.pClientData[iTermId].dbproc);
    }

    UNUSEDPARAM(iSyncId);
}

/* FUNCTION: BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS **dbproc, char *server, char *database,
char *user, char *password, char *app, int *spid, long *pack_size)
*
* PURPOSE: This function opens the sql connection for use.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK *pECB passed in structure pointer from inetsrv.
* int iTermId terminal id of browser
* int iSyncId sync id of browser
* DBPROCESS **dbproc pointer to returned DBPROCESS
* char *server SQL server name
* char *database SQL server database
* char *user user name
* char *password user password
* char *app pointer to returned application array
* int *spid pointer to returned spid
* long *pack_size pointer to returned default pack size
*
* RETURNS: BOOL FALSE if successfull
* TRUE if an error occurs
*
* COMMENTS: None
*/

#ifdef USE_ODBC
static BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS **dbproc, char *server, char *database,
char *user, char *password, char *app, int *spid, long *pack_size)
{
    RETCODE rc;
    char buffer[30];

    *dbproc = (DBPROCESS *)malloc(sizeof(DBPROCESS));
    if ( !*dbproc )
        return TRUE;

    //set pECB data into dbproc
    (*dbproc)->bDeadlock = FALSE;
    (*dbproc)->bFailed = FALSE;
    (*dbproc)->pECB = pECB;
    (*dbproc)->iTermId = iTermId;
    (*dbproc)->iSyncId = iSyncId;

    if ( SQLAllocConnect(henv, &(*dbproc)->hdbc) == SQL_ERROR )
        return TRUE;

    if ( SQLSetConnectOption((*dbproc)->hdbc, SQL_PACKET_SIZE, pack_size) == SQL_ERROR )
        return TRUE;

    rc = SQLConnect((*dbproc)->hdbc, server, SQL_NTS, user, SQL_NTS, password, SQL_NTS);
    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
        return TRUE;
    rc = SQLAllocStmt((*dbproc)->hdbc, &(*dbproc)->hstmt);
    if (rc == SQL_ERROR)
        return TRUE;

    sprintf(buffer, "use %s", Client->database);

    rc = SQLExecDirect((*dbproc)->hstmt, buffer, SQL_NTS);
    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
        return TRUE;

    SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);
    sprintf(buffer, "set nocount on");
    rc = SQLExecDirect((*dbproc)->hstmt, buffer, SQL_NTS);
    if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
        return TRUE;
    SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);
}
#endif
```

## Appendix A - Application Source Code

---

```
        sprintf(buffer,"select @@spid");

        rc = SQLExecDirect((*dbproc)->hstmt, buffer, SQL_NTS);
        if (rc != SQL_SUCCESS && rc != SQL_SUCCESS_WITH_INFO)
            return TRUE;

        if ( SQLBindCol((*dbproc)->hstmt, 1, SQL_C_SSHORT, &(*dbproc)->spid, 0, NULL) == SQL_ERROR )
            return TRUE;

        if ( SQLFetch((*dbproc)->hstmt) == SQL_ERROR )
            return TRUE;

        SQLFreeStmt((*dbproc)->hstmt, SQL_CLOSE);

        return FALSE;
    }

#else

static BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS **dbproc, char *server, char *database,
char *user, char *password, char *app, int *spid)
{
    LOGINREC  *login;
    PECBINFO  pEcbInfo;

    //set local msg proc for login record
    //attach pECB record

    //this is necessary as dblib provides no way to pass user data in a login structure. So until
    //there is an allocated dbproc we need to use a static which means that the login attempt must
    //be serialized.

    gpECB = pECB;

    login = dblogin();
    if ( !*user )
        DBSETLUSER(login, "sa");
    else
        DBSETLUSER(login, user);

    DBSETLPWD(login, password);
    DBSETLHOST(login, app);

    DBSETLPACKET(login, (unsigned short)DEFCLPACKSIZE);

    if ((*dbproc = dbopen(login, server )) == NULL)
        return TRUE;

    //set pECB data into dbproc
    pEcbInfo = (PECBINFO)malloc(sizeof(PECBINFO));
    pEcbInfo->bDeadlock = FALSE;
    pEcbInfo->pECB = pECB;
    pEcbInfo->iTermId = iTermId;
    pEcbInfo->iSyncId = iSyncId;
    dbsetuserdata(*dbproc, pEcbInfo);

    // Use the the right database
    dbuse(*dbproc, database);

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

    dbsqlxec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    {
        dbbind(*dbproc, 1, SMALLBIND, (DBINT) 0, (BYTE *) spid);
        while (dbnextrow(*dbproc) != NO_MORE_ROWS)
            ;
    }
    dbcmd(*dbproc, "set nocount on");

    dbsqlxec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    {
        while (dbnextrow(*dbproc) != NO_MORE_ROWS)
            ;
    }

    //rollback transaction on abort
    dbcmd(*dbproc, "set XACT_ABORT ON");

    dbsqlxec(*dbproc);
    while (dbresults(*dbproc) != NO_MORE_RESULTS)
    {
        while (dbnextrow(*dbproc) != NO_MORE_ROWS)
            ;
    }

    return FALSE;
}
}
```

## Appendix A - Application Source Code

---

```
#endif

/* FUNCTION: BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS *dbproc)
 *
 * PURPOSE: This function closes the sql connection.
 *
 * ARGUMENTS:      EXTENSION_CONTROL_BLOCK *pECB      passed in structure pointer from inetsrv.
 *                  DBPROCESS *dbproc                pointer to DBPROCESS
 *
 * RETURNS:        BOOL      FALSE      if successfull
 *                  TRUE      if an error occurs
 *
 * COMMENTS:       None
 */

#ifdef USE_ODBC
static BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS *dbproc)
{
    if ( dbproc )
    {
        SQLFreeStmt(dbproc->hstmt, SQL_DROP);
        SQLDisconnect(dbproc->hdbc);
        SQLFreeConnect(dbproc->hdbc);
        free(dbproc);
        dbproc = NULL;
    }
    return FALSE;
}
#else
static BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS *dbproc)
{
    if (dbclose(dbproc) == FAIL)
        return TRUE;
    return FALSE;
}
#endif

/* FUNCTION: SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry)
 *
 * PURPOSE: This function handles the stock level transaction.
 *
 * ARGUMENTS:      EXTENSION_CONTROL_BLOCK *pECB      passed in structure pointer from inetsrv.
 *                  int iTermId                       terminal id of browser
 *                  int iSyncId                       sync id of browser
 *                  DBPROCESS *dbproc                 connection db
 *                  STOCK_LEVEL_DATA *pStockLevel     stock level input / output data structure
 *                  short deadlock_retry              deadlock_retry count if deadlocked
 *
 * RETURNS:        BOOL      FALSE      if successfull
 *                  TRUE      if deadlocked
 *
 * COMMENTS:       None
 */

static BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry)
{
    int rc;
    RETCODE rc;
    char printbu[25];
    BYTE *pData;
    PECBINFO pEcbInfo;

    //update pECB and bFailed flag
    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
        pEcbInfo->pECB = pECB;
        pEcbInfo->bFailed = FALSE;
        pEcbInfo->iTermId = iTermId;
        pEcbInfo->iSyncId = iSyncId;
    }

    pStockLevel->num_deadlocks = 0;

    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
        if (dbrpcinit(dbproc, "tpcc_stocklevel", 0) == SUCCEED)
        {
            dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pStockLevel->w_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pStockLevel->d_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pStockLevel->thresh_hold);

            if (dbrpcexec(dbproc) == SUCCEED)
            {
                while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
            }
        }
    }
}

```

## Appendix A - Application Source Code

---

```
        {
            if (DBROWS(dbproc))
            {
                while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
                {
                    if (pData=dbdata(dbproc, 1))
                        pStockLevel->low_stock = *((long *) pData);
                }
            }
        }
    }
    if (SQLDetectDeadlock(dbproc))
    {
        pStockLevel->num_deadlocks++;
        sprintf(printbuf,"deadlock: retry: %d",pStockLevel->num_deadlocks);
        Sleep(10 * tryit);
    }
    else
    {
        strcpy(pStockLevel->execution_status, "Transaction committed.");
        return FALSE;
    }
}

// If we reached here, it means we quit after MAX_RETRY deadlocks
strcpy(pStockLevel->execution_status, "Hit deadlock max. ");
return TRUE;
}

/* FUNCTION: int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, int iTermId, int iSyncId, DBPROCESS *dbproc, NEW_ORDER_DATA
*pNewOrder, short deadlock_retry)
*
* PURPOSE: This function handles the new order transaction.
*
* ARGUMENTS:      EXTENSION_CONTROL_BLOCK      *pECB      passed in structure pointer from inetsrv.
*                  int                          iTermId
*                  terminal id of browser
*                  int                          iSyncId
*                  sync id of browser
*                  DBPROCESS                     *dbproc      connection db
process id
*                  NEW_ORDER_DATA               *pNewOrder   pointer to new order structure
for input/output data
*                  short                        deadlock_retry  count if deadlocked
*
* RETURNS:        int      TRUE      transaction committed
*                  FALSE      item number not valid
*                  -1         deadlock max retry reached
*
* COMMENTS:      None
*/

static int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, NEW_ORDER_DATA *pNewOrder, short
deadlock_retry)
{
    RETCODE          rc;
    int              i;
    DBINT           commit_flag;
    int             tryit;
    char            printbuf[25];
    char            tmpbuf[30];
    DBDATETIME      datetime;
    BYTE            *pData;
    PECBINFO        pEcbInfo;

    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
        pEcbInfo->pECB = pECB;
        pEcbInfo->bFailed = FALSE;
        pEcbInfo->iTermId = iTermId;
        pEcbInfo->iSyncId = iSyncId;
    }

    pNewOrder->num_deadlocks = 0;

    strcpy(tmpbuf, "tpcc_neworder");

    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
        if (dbrcpinit(dbproc, tmpbuf, 0) == SUCCEED)
        {
            dbrcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pNewOrder->w_id);
            dbrcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder->d_id);
            dbrcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pNewOrder->c_id);
            dbrcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder->o_ol_cnt);
            dbrcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pNewOrder->o_all_local);

            pNewOrder->o_all_local = 1;
        }
    }
}
```



## Appendix A - Application Source Code

```
for (i = 0; i < pNewOrder->o_ol_cnt; i++)
{
    if ( pNewOrder->o_all_local && pNewOrder->Ol[i].ol_supply_w_id != pNewOrder->w_id )
        pNewOrder->o_all_local = 0;
    dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pNewOrder->Ol[i].ol_i_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pNewOrder->Ol[i].ol_supply_w_id);
    dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pNewOrder->Ol[i].ol_quantity);
}

if (dbrpcexec(dbproc) == SUCCEEDED)
{
    pNewOrder->total_amount=0;

    // Get results from order line
    for (i = 0; i<pNewOrder->o_ol_cnt; i++)
    {
        if (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
        {
            if (DBROWS(dbproc) && (dbnumcols(dbproc) == 5))
            {
                while (dbnextrow(dbproc) != NO_MORE_ROWS)
                {
                    if(pData=dbdata(dbproc, 1))
                        UtilStrCpy(pNewOrder->Ol[i].ol_i_name,

                    if(pData=dbdata(dbproc, 2))
                        pNewOrder->Ol[i].ol_stock =

                    if(pData=dbdata(dbproc, 3))
                        UtilStrCpy(pNewOrder-

                    if(pData=dbdata(dbproc, 4))
                        pNewOrder->Ol[i].ol_i_price = *(DBFLT8

                    if(pData=dbdata(dbproc, 5))

                        pNewOrder->Ol[i].ol_amount = *(DBFLT8

                    pNewOrder->total_amount = pNewOrder->total_amount +

                }
            }
        }
        while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
        {
            if (DBROWS(dbproc) && (dbnumcols(dbproc) == 8))
            {
                while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
                {
                    if(pData=dbdata(dbproc, 1))

                        pNewOrder->w_tax = *(DBFLT8 *) pData);

                    if(pData=dbdata(dbproc, 2))

                        pNewOrder->d_tax = *(DBFLT8 *) pData);

                    if(pData=dbdata(dbproc, 3))
                        pNewOrder->o_id = *(DBINT *) pData);
                    if(pData=dbdata(dbproc, 4))
                        UtilStrCpy(pNewOrder->c_last, pData, dbdatlen(dbproc,

                    if(pData=dbdata(dbproc, 5))
                        pNewOrder->c_discount = *(DBFLT8 *) pData);

                    if(pData=dbdata(dbproc, 6))
                        UtilStrCpy(pNewOrder->c_credit, pData, dbdatlen(dbproc,

                    if(pData=dbdata(dbproc, 7))
                    {
                        datetime = *(DBDATETIME *) pData);
                        dbdatecrack(dbproc, &pNewOrder->o_entry_d,

                    }
                    if(pData=dbdata(dbproc, 8))commit_flag = *(DBTINYINT *) pData);
                }
            }
        }
    }
}
```

## Appendix A - Application Source Code

---

```
    }
    if (SQLDetectDeadlock(dbproc))
    {
        pNewOrder->num_deadlocks++;
        sprintf(printbuf,"deadlock: retry: %d",pNewOrder->num_deadlocks);
        Sleep(DEADLOCKWAIT*tryit);
    }
    else
    {
        if (commit_flag == 1)
        {
            pNewOrder->total_amount = pNewOrder->total_amount * ((1 + pNewOrder->w_tax + pNewOrder->d_tax) * (1 -
pNewOrder->c_discount));
            strcpy(pNewOrder->execution_status,"Transaction committed.");
            return TRUE;
        }
        else
        {
            strcpy(pNewOrder->execution_status,"Item number is not valid.");
            return FALSE;
        }
    }
}

// If we reached here, it means we quit after MAX_RETRY deadlocks
strcpy(pNewOrder->execution_status,"Hit deadlock max. ");

return -1; // "deadlock max retry reached!"
}

/* FUNCTION: int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short
deadlock_retry)
*
* PURPOSE: This function handles the payment transaction.
*
* ARGUMENTS:      EXTENSION_CONTROL_BLOCK      *pECB                passed in structure pointer from inetsrv.
*                  int                          iTermId
*                  int                          iSyncId
*                  DBPROCESS                    *dbproc              connection db
process id
*                  PAYMENT_DATA                *pPayment            pointer to payment
input/output data structure
*                  short                        deadlock_retry        deadlock retry count
*
* RETURNS:        int      TRUE                success
*                  -1                max deadlocked reached
*
* COMMENTS:      None
*
*/

static int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short deadlock_retry)
{
    RETCODE          rc;
    int              tryit;
    char              printbuf[26];
    BOOL              by_name;
    DBDATETIME       datetime;
    BYTE              *pData;
    PECBINFO         pEcbInfo;

    if ((pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
        pEcbInfo->pECB = pECB;
        pEcbInfo->bFailed = FALSE;
        pEcbInfo->iTermId = iTermId;
        pEcbInfo->iSyncId = iSyncId;
    }

    pPayment->num_deadlocks = 0;

    if (pPayment->c_id == 0)
        by_name = TRUE;
    else
        by_name = FALSE;

    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
        if (dbrpcinit(dbproc, "tpcc_payment", 0) == SUCCEEDED)
        {
            dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pPayment->w_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pPayment->c_w_id);
            dbrpcparam(dbproc, NULL, 0, SQLFLT8, -1, -1, (BYTE *) &pPayment->h_amount);
            dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pPayment->d_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pPayment->c_d_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pPayment->c_id);
            if (pPayment->c_id == 0)
                {
```

## Appendix A - Application Source Code

```
        dbrcparam(dbproc, NULL, 0, SQLCHAR, -1, strlen(pPayment->c_last), pPayment->c_last);
    }
}
if (dbrcexec(dbproc) == SUCCEEDED)
{
    while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
    {
        if (DBROWS(dbproc) && (dbnumcols(dbproc) == 27))
        {
            while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
            {
                if(pData=dbdata(dbproc, 1))
                    pPayment->c_id = *((DBINT *) pData);
                if(pData=dbdata(dbproc, 2))
                    UtilStrCpy(pPayment->c_last, pData, dbdatlen(dbproc, 2));
                if(pData=dbdata(dbproc, 3))
                {
                    datetime = *((DBDATETIME *) pData);
                    dbdatecrack(dbproc, &pPayment->h_date, &datetime);
                }
                if(pData=dbdata(dbproc, 4))
                    UtilStrCpy(pPayment->w_street_1, pData, dbdatlen(dbproc, 4));
                if(pData=dbdata(dbproc, 5))
                    UtilStrCpy(pPayment->w_street_2, pData, dbdatlen(dbproc, 5));
                if(pData=dbdata(dbproc, 6))
                    UtilStrCpy(pPayment->w_city, pData, dbdatlen(dbproc, 6));
                if(pData=dbdata(dbproc, 7))
                    UtilStrCpy(pPayment->w_state, pData, dbdatlen(dbproc, 7));
                if(pData=dbdata(dbproc, 8))
                    UtilStrCpy(pPayment->w_zip, pData, dbdatlen(dbproc, 8));
                if(pData=dbdata(dbproc, 9))
                    UtilStrCpy(pPayment->d_street_1, pData, dbdatlen(dbproc, 9));
                if(pData=dbdata(dbproc, 10))
                    UtilStrCpy(pPayment->d_street_2, pData, dbdatlen(dbproc, 10));
                if(pData=dbdata(dbproc, 11))
                    UtilStrCpy(pPayment->d_city, pData, dbdatlen(dbproc, 11));
                if(pData=dbdata(dbproc, 12))
                    UtilStrCpy(pPayment->d_state, pData, dbdatlen(dbproc, 12));
                if(pData=dbdata(dbproc, 13))
                    UtilStrCpy(pPayment->d_zip, pData, dbdatlen(dbproc, 13));
                if(pData=dbdata(dbproc, 14))
                    UtilStrCpy(pPayment->c_first, pData, dbdatlen(dbproc, 14));
                if(pData=dbdata(dbproc, 15))
                    UtilStrCpy(pPayment->c_middle, pData, dbdatlen(dbproc, 15));
                if(pData=dbdata(dbproc, 16))
                    UtilStrCpy(pPayment->c_street_1, pData, dbdatlen(dbproc, 16));
                if(pData=dbdata(dbproc, 17))
                    UtilStrCpy(pPayment->c_street_2, pData, dbdatlen(dbproc, 17));
                if(pData=dbdata(dbproc, 18))
                    UtilStrCpy(pPayment->c_city, pData, dbdatlen(dbproc, 18));
                if(pData=dbdata(dbproc, 19))
                    UtilStrCpy(pPayment->c_state, pData, dbdatlen(dbproc, 19));
                if(pData=dbdata(dbproc, 20))
                    UtilStrCpy(pPayment->c_zip, pData, dbdatlen(dbproc, 20));
                if(pData=dbdata(dbproc, 21))
                    UtilStrCpy(pPayment->c_phone, pData, dbdatlen(dbproc, 21));
                if(pData=dbdata(dbproc, 22))
                {
                    datetime = *((DBDATETIME *) pData);
                    dbdatecrack(dbproc, &pPayment->c_since, &datetime);
                }
                if(pData=dbdata(dbproc, 23))
                    UtilStrCpy(pPayment->c_credit, pData, dbdatlen(dbproc, 23));
                if(pData=dbdata(dbproc, 24))
                    pPayment->c_credit_lim = *((DBFLT8 *) pData);
                if(pData=dbdata(dbproc, 25))
                    pPayment->c_discount = *((DBFLT8 *) pData);
                if(pData=dbdata(dbproc, 26))
                    pPayment->c_balance = *((DBFLT8 *) pData);
                if(pData=dbdata(dbproc, 27))
                    UtilStrCpy(pPayment->c_data, pData, dbdatlen(dbproc, 27));
            }
        }
    }
}
if (SQLDetectDeadlock(dbproc))
{
    pPayment->num_deadlocks++;
    sprintf(printbuf, "deadlock: retry: %d", pPayment->num_deadlocks);
    Sleep(DEADLOCKWAIT*tryit);
}
else
{
    if ( pPayment->c_id == 0 )
    {
        strcpy(pPayment->execution_status, "Invalid Customer id.name.");
        return 0;
    }
    else
        strcpy(pPayment->execution_status, "Transaction committed.");
    return TRUE;
}
}
```

## Appendix A - Application Source Code

---

```
    }
    // If we reached here, it means we quit after MAX_RETRY deadlocks
    strcpy(pPayment->execution_status,"Hit deadlock max. ");
    return -1; /*"deadlock max retry reached!"
}

/* FUNCTION: int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus,
short deadlock_retry)
*
* PURPOSE: This function processes the Order Status transaction.
*
* ARGUMENTS:      EXTENSION_CONTROL_BLOCK    *pECB                passed in structure pointer from inetsrv.
                  int                        iTermId
                  terminal id of browser
                  int                        iSyncId
                  sync id of browser
                  DBPROCESS                  *dbproc                connection db
process id
*
*                  ORDER_STATUS_DATA        *pOrderStatus pointer to Order Status data input/output structure
                  short                    deadlock_retry          deadlock_retry count
*
* RETURNS:        int        -1                max deadlock reached
                  int        0                No orders found for customer
                  int        1                Transaction successfull
*
* COMMENTS:      None
*
*/

static int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId, DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus, short
deadlock_retry)
{
    RETCODE          rc;
    int              tryit;
    int              i;
    char             printbuf[25];
    BOOL             by_name;
    DBDATETIME      datetime;
    BYTE             *pData;
    PECBINFO        pEcbInfo;

    if ( ( pEcbInfo = (PECBINFO)dbgetuserdata(dbproc) ) )
    {
        pEcbInfo->pECB = pECB;
        pEcbInfo->bFailed = FALSE;
        pEcbInfo->iTermId = iTermId;
        pEcbInfo->iSyncId = iSyncId;
    }

    pOrderStatus->num_deadlocks = 0;
    if (pOrderStatus->c_id == 0)
        by_name = TRUE;
    else
        by_name = FALSE;

    for (tryit=0; tryit < deadlock_retry; tryit++)
    {
        if (dbrcpinit(dbproc, "tpcc_orderstatus", 0) == SUCCEED)
        {
            dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE *) &pOrderStatus->w_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE *) &pOrderStatus->d_id);
            dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE *) &pOrderStatus->c_id);
            if (pOrderStatus->c_id == 0)
            {
                dbrpcparam(dbproc, NULL, 0, SQLCHAR, -1, strlen(pOrderStatus->c_last), pOrderStatus->c_last);
            }
        }
        if (dbrcpexec(dbproc) == SUCCEED)
        {
            while (((rc = dbresults(dbproc)) != NO_MORE_RESULTS) && (rc != FAIL))
            {
                if (DBROWS(dbproc) && (dbnumcols(dbproc) == 5))
                {
                    i=0;
                    while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
                    {
                        if(pData=dbdata(dbproc, 1))
                            pOrderStatus->OIOrderStatusData[i].ol_supply_w_id =
                                (*DBSMALLINT *) pData;
                        if(pData=dbdata(dbproc, 2))
                            pOrderStatus->OIOrderStatusData[i].ol_i_id = (*(DBINT *) pData);
                        if(pData=dbdata(dbproc, 3))
                            pOrderStatus->OIOrderStatusData[i].ol_quantity = (*(DBSMALLINT *)
                                pData);
                        if(pData=dbdata(dbproc, 4))
                            pOrderStatus->OIOrderStatusData[i].ol_amount = (*(DBFLT8 *) pData);
                        if(pData=dbdata(dbproc, 5))
                        {
                            datetime = (*(DBDATETIME *) pData);
                        }
                    }
                }
            }
        }
    }
}
```

## Appendix A - Application Source Code

```

                                                                 dbdatecrack(dbproc, &pOrderStatus->OIOrderStatusData[i].ol_delivery_d,
&datetime);
    }
    }
    pOrderStatus->o_ol_cnt = i;
}
else if (DBROWS(dbproc) && (dbnumcols(dbproc) == 8))
{
    while (((rc = dbnextrow(dbproc)) != NO_MORE_ROWS) && (rc != FAIL))
    {
        if(pData=dbdata(dbproc, 1))
            pOrderStatus->c_id = (*(DBINT *) pData);
        if(pData=dbdata(dbproc, 2))
            UtilStrCpy(pOrderStatus->c_last, pData, dbdatlen(dbproc,2));
        if(pData=dbdata(dbproc, 3))
            UtilStrCpy(pOrderStatus->c_first, pData, dbdatlen(dbproc,3));
        if(pData=dbdata(dbproc, 4))
            UtilStrCpy(pOrderStatus->c_middle, pData, dbdatlen(dbproc, 4));
        if(pData=dbdata(dbproc, 5))
        {
            datetime = (*(DBDATETIME *) pData);
            dbdatecrack(dbproc, &pOrderStatus->o_entry_d, &datetime);
        }
        if(pData=dbdata(dbproc, 6))
            pOrderStatus->o_carrier_id = (*(DBSMALLINT *) pData);
        if(pData=dbdata(dbproc, 7))
            pOrderStatus->c_balance = (*(DBFLT8 *) pData);
        if(pData=dbdata(dbproc, 8))
            pOrderStatus->o_id = (*(DBINT *) pData);
    }
}
if (i==0)
    return 0; /*"No orders found for customer"
}
}
if (SQLDetectDeadlock(dbproc))
{
    pOrderStatus->num_deadlocks++;
    sprintf(printbuf,"deadlock: retry: %d",pOrderStatus->num_deadlocks);
    Sleep(DEADLOCKWAIT*tryit);
}
else
{
    if (pOrderStatus->c_id == 0 && pOrderStatus->c_last[0] == 0)
        strcpy(pOrderStatus->execution_status,"Invalid Customer id.name.");
    else
        strepy(pOrderStatus->execution_status,"Transaction committed.");
    return 1;
}
}
// If we reached here, it means we quit after MAX_RETRY deadlocks
strepy(pOrderStatus->execution_status,"Hit deadlock max. ");
return -1; /*"deadlock max retry reached!"
}

/* FUNCTION: BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
*
* PURPOSE: This function checks to see if a sql server deadlock condition exists.
*
* ARGUMENTS:      DBPROCESS          *dbproc          connection db process id to check
*
* RETURNS:        BOOL      FALSE          no deadlock detected
*                  TRUE          deadlock condition exists
*
* COMMENTS:       None
*
*/

BOOL SQLDetectDeadlock(DBPROCESS *dbproc)
{
    PECBINFO  pEcbInfo;

    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(dbproc)) )
    {
        if ( pEcbInfo->bDeadlock )
        {
            pEcbInfo->bDeadlock = FALSE;
            return TRUE;
        }
    }
    return FALSE;
}

/* FUNCTION: void FormatString(char *szDest, char *szPic, char *szSrc)
*
* PURPOSE: This function formats a character string for inclusion in the
*          HTML formatted page being constructed.
*
* ARGUMENTS:      char          *szDest      Destination buffer where formatted string is to be placed
*                  char          *szPic      picture string which describes how character value is to be

```

## Appendix A - Application Source Code

---

```
*
*                                     char          *szSrc      formatted.
*                                     character string value.
*
* RETURNS:          None
*
* COMMENTS:        This functions is used to format TPC-C phone and zip value strings.
*
*/

static void FormatString(char *szDest, char *szPic, char *szSrc)
{
    while( *szPic )
    {
        if ( *szPic == 'X' )
        {
            if ( *szSrc )
                *szDest++ = *szSrc++;
            else
                *szDest++ = ' ';
        }
        else
            *szDest++ = *szPic;
        szPic++;
    }
    *szDest = 0;

    return;
}

/* FUNCTION: char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL bInput)
*
* PURPOSE: This function constructs the Stock Level HTML page.
*
* ARGUMENTS:      int          iTermId      client browser terminal id
*                  int          iSyncId     client browser sync id
*                  BOOL         bInput      TRUE if form is being constructed for input else FALSE
*
* RETURNS:        char *                A pointer to buffer inside client structure where HTML form is built.
*
* 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.
*/

static char *MakeStockLevelForm(int iTermId, int iSyncId, BOOL bInput)
{
    char          *szForm;

    szForm = (char *)Term.pClientData[iTermId].szBuffer;

    Term.pClientData[iTermId].stockLevelData.w_id          = (short)Term.pClientData[iTermId].w_id;
    Term.pClientData[iTermId].stockLevelData.d_id         = (short)Term.pClientData[iTermId].d_id;
    Term.pClientData[iTermId].stockLevelData.num_deadlocks = 0;

    strcpy(szForm, "<HTML><HEAD><TITLE>TPC-C Stock Level</TITLE></HEAD>");
    strcat(szForm, "<FORM ACTION=\"tpcc.dll\" METHOD=\"GET\">");
    if ( bInput )
        strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"PI\" VALUE=\"1\">");
    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">");
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">", STOCK_LEVEL_FORM);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"TERMINID\" VALUE=\"%d\">", iTermId);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"SYNCID\" VALUE=\"%d\">", iSyncId);
    strcat(szForm, "<PRE>
        Stock-Level<BR>");
    sprintf(szForm+strlen(szForm), "Warehouse: %4.4d District: %2.2d<BR><BR>", Term.pClientData[iTermId].stockLevelData.w_id,
    Term.pClientData[iTermId].stockLevelData.d_id);
    if ( bInput )
    {
        strcat(szForm, "Stock Level Threshold: <INPUT NAME=\"TT*\" SIZE=2<BR><BR>
            \"low stock: <BR><HR>
            \"<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\">
            \"<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">");
    }
    else
    {
        sprintf(szForm+strlen(szForm), "Stock Level Threshold: %2.2d<BR><BR>", Term.pClientData[iTermId].stockLevelData.thresh_hold);

        sprintf(szForm+strlen(szForm), "low stock: %3.3d<PRE><BR><HR>", Term.pClientData[iTermId].stockLevelData.low_stock);
        strcat(szForm, "<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..\">");
    }

    strcat(szForm, "</FORM></HTML>");

    return szForm;
}

/* FUNCTION: char *MakeMainMenuForm(int iTermId, int iSyncId)
*
*/
```

## Appendix A - Application Source Code

---

```
* PURPOSE: This function
*
* ARGUMENTS:      int                int                iTermId    client browser terminal id
*                int                iSyncId           client browser sync id
*
* RETURNS:        char *              A pointer to buffer inside client structure where HTML form is built.
*
* 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.
*/

static char *MakeMainMenuForm(int iTermId, int iSyncId)
{
    char        *szForm;

    szForm = (char *)Term.pClientData[iTermId].szBuffer;

    strcpy(szForm,
           "<HTML><HEAD><TITLE>TPC-C Main Menu</TITLE></HEAD><BODY>"
           "Select Desired Transaction.<BR><HR>"
           "<FORM ACTION=\\\"tpcc.dll\\\" METHOD=\\\"GET\\\">");
    strcat(szForm, "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"STATUSID\\\" VALUE=\\\"0\\\">");
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"TERMINID\\\" VALUE=\\\"%d\\\">", iTermId);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"SYNCID\\\" VALUE=\\\"%d\\\">", iSyncId);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"FORMID\\\" VALUE=\\\"%d\\\">", MAIN_MENU_FORM);
    strcat(szForm,
           "<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>");

    return szForm;
}

/* FUNCTION: char *MakeWelcomeForm(void)
*
* PURPOSE: This function
*
* ARGUMENTS: None
*
* RETURNS:      char *              A pointer to the static HTML welcome form.
*
* COMMENTS:     The welcome form is static.
*/

static char *MakeWelcomeForm(void)
{
    return szWelcomeForm;
}

/* FUNCTION: char *MakeNewOrderForm(int iTermId, BOOL bInput, BOOL bValid)
*
* PURPOSE: This function
*
* ARGUMENTS:      int                int                iTermId    client browser terminal id
*                int                iSyncId           client browser sync id
*                BOOL               bInput           TRUE if form is being constructed for input else FALSE
*                BOOL               bValid           TRUE if NeworderData valid, ELSE FALSE effects output
*
* RETURNS:        char *              A pointer to buffer inside client structure where HTML form is built.
*
* 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.
*/

static char *MakeNewOrderForm(int iTermId, int iSyncId, BOOL bInput, BOOL bValid)
{
    char        *szForm;
    char        szName[146];
    char        szCredit[14];
    int         i;

    szForm = (char *)Term.pClientData[iTermId].szBuffer;

    Term.pClientData[iTermId].newOrderData.w_id = Term.pClientData[iTermId].w_id;

    strcpy(szForm, "<HTML>"
           "<HEAD><TITLE>TPC-C New Order</TITLE></HEAD><BODY>"
           "<FORM ACTION=\\\"tpcc.dll\\\" METHOD=\\\"GET\\\">");

    if ( bInput )
        strcat(szForm, "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"PI*\\\" VALUE=\\\"\\\">");

    strcat(szForm, "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"STATUSID\\\" VALUE=\\\"0\\\">");
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"FORMID\\\" VALUE=\\\"%d\\\">", NEW_ORDER_FORM);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"TERMINID\\\" VALUE=\\\"%d\\\">", iTermId);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\\\"hidden\\\" NAME=\\\"SYNCID\\\" VALUE=\\\"%d\\\">", iSyncId);
    strcat(szForm, "<PRE>
    New Order<BR>");
}
```

## Appendix A - Application Source Code

```
        if ( bInput )
        {
            sprintf(szForm+strlen(szForm), "Warehouse: %4.4d District: <INPUT NAME=\"DID*\" SIZE=1>          Date:<BR>",
Term.pClientData[iTermId].newOrderData.w_id);
            strcat(szForm, "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=\"SP01*\" SIZE=4> <INPUT NAME=\"IID01*\" SIZE=6>"
                " <INPUT NAME=\"SP02*\" SIZE=4> <INPUT NAME=\"IID02*\" SIZE=6>"
                " <INPUT NAME=\"SP03*\" SIZE=4> <INPUT NAME=\"IID03*\" SIZE=6>"
                " <INPUT NAME=\"SP04*\" SIZE=4> <INPUT NAME=\"IID04*\" SIZE=6>"
                " <INPUT NAME=\"SP05*\" SIZE=4> <INPUT NAME=\"IID05*\" SIZE=6>"
                " <INPUT NAME=\"SP06*\" SIZE=4> <INPUT NAME=\"IID06*\" SIZE=6>"
                " <INPUT NAME=\"SP07*\" SIZE=4> <INPUT NAME=\"IID07*\" SIZE=6>"
                " <INPUT NAME=\"SP08*\" SIZE=4> <INPUT NAME=\"IID08*\" SIZE=6>"
                " <INPUT NAME=\"SP09*\" SIZE=4> <INPUT NAME=\"IID09*\" SIZE=6>"
                " <INPUT NAME=\"SP10*\" SIZE=4> <INPUT NAME=\"IID10*\" SIZE=6>"
                " <INPUT NAME=\"SP11*\" SIZE=4> <INPUT NAME=\"IID11*\" SIZE=6>"
                " <INPUT NAME=\"SP12*\" SIZE=4> <INPUT NAME=\"IID12*\" SIZE=6>"
                " <INPUT NAME=\"SP13*\" SIZE=4> <INPUT NAME=\"IID13*\" SIZE=6>"
                " <INPUT NAME=\"SP14*\" SIZE=4> <INPUT NAME=\"IID14*\" SIZE=6>"
                "Execution Status:          Total:<BR><HR>"
                "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\">"
                "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">"
                "<FORM>"
                "</HTML>");
        }
    }
    else
    {
        if ( bValid )
        {
            sprintf(szForm+strlen(szForm), "Warehouse: %4.4d District: %2.2d          Date: %2.2d-%2.2d-%4.4d
%2.2d:%2.2d:%2.2d <BR>",
Term.pClientData[iTermId].newOrderData.w_id,
Term.pClientData[iTermId].newOrderData.d_id,
Term.pClientData[iTermId].newOrderData.o_entry_d.day,
Term.pClientData[iTermId].newOrderData.o_entry_d.month,
Term.pClientData[iTermId].newOrderData.o_entry_d.year,
Term.pClientData[iTermId].newOrderData.o_entry_d.hour,
Term.pClientData[iTermId].newOrderData.o_entry_d.minute,
Term.pClientData[iTermId].newOrderData.o_entry_d.second);
        }
        else
        {
            sprintf(szForm+strlen(szForm), "Warehouse: %4.4d District: %2.2d          Date:<BR>",
Term.pClientData[iTermId].newOrderData.w_id,
Term.pClientData[iTermId].newOrderData.d_id);
        }
        FormatHTMLString(szName, Term.pClientData[iTermId].newOrderData.c_last, 16);
        FormatHTMLString(szCredit, Term.pClientData[iTermId].newOrderData.c_credit, 2);
        sprintf(szForm+strlen(szForm), "Customer: %4.4d Name: %s Credit: %s ",
Term.pClientData[iTermId].newOrderData.c_id, szName, szCredit);
        if ( bValid )
        {
            sprintf(szForm+strlen(szForm), "%Disc: %5.2f          <BR>", Term.pClientData[iTermId].newOrderData.c_discount);
            sprintf(szForm+strlen(szForm), "Order Number: %8.8d Number of Lines: %2.2d      W_tax: %5.2f      D_tax: %5.2f <BR><BR>",
Term.pClientData[iTermId].newOrderData.o_id,
Term.pClientData[iTermId].newOrderData.o.ol_cnt,
Term.pClientData[iTermId].newOrderData.w_tax,
Term.pClientData[iTermId].newOrderData.d_tax);
            strcat(szForm, " Supp_W Item_Id Item Name      Qty Stock B/G Price Amount<BR>");
            for(i=0; i<Term.pClientData[iTermId].newOrderData.ol_cnt; i++)
            {
                FormatHTMLString(szName, Term.pClientData[iTermId].newOrderData.Ol[i].ol_i_name, 24);
                sprintf(szForm+strlen(szForm), " %4.4d %6.6d %s %2.2d %3.3d %1.1s %6.2f %7.2f <BR>",
Term.pClientData[iTermId].newOrderData.Ol[i].ol_supply_w_id,
Term.pClientData[iTermId].newOrderData.Ol[i].ol_i_id,
```



## Appendix A - Application Source Code

```
        szName,
        Term.pClientData[iTermId].newOrderData.Ol[i].ol_quantity,
        Term.pClientData[iTermId].newOrderData.Ol[i].ol_stock,
        Term.pClientData[iTermId].newOrderData.Ol[i].ol_brand_generic,
        Term.pClientData[iTermId].newOrderData.Ol[i].ol_i_price,
        Term.pClientData[iTermId].newOrderData.Ol[i].ol_amount );
    }
}
else
{
    strcat(szForm, "%Disc:<BR>");
    sprintf(szForm+strlen(szForm), "Order Number: %8.8d Number of Lines:      W_tax:      D_tax:<BR><BR>",
        Term.pClientData[iTermId].newOrderData.o_id);

    strcat(szForm, " Supp_W Item_Id Item Name      Qty Stock B/G Price  Amount<BR>");

    i = 0;
}
for( i<15; i++)
    strcat(szForm, "<BR>");

if ( bValid )
{
    sprintf(szForm+strlen(szForm), "Execution Status: %24.24s      Total: $%8.2f ",
        Term.pClientData[iTermId].newOrderData.execution_status,
        Term.pClientData[iTermId].newOrderData.total_amount);
}
else
{
    sprintf(szForm+strlen(szForm), "Execution Status: %24.24s      Total:",
        Term.pClientData[iTermId].newOrderData.execution_status);
}

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

        "<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..'>");

strcat(szForm, "</FORM></HTML>");
}

return szForm;
}

/* FUNCTION: char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput)
 *
 * PURPOSE: This function
 *
 * ARGUMENTS:      int          iTermId      client browser terminal id
 *                  int          iSyncId      client browser sync id
 *                  BOOL         bInput       TRUE if form is being constructed for input else FALSE
 *
 * RETURNS:        char *          A pointer to buffer inside client structure where HTML form is built.
 *
 * 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.
 */

static char *MakePaymentForm(int iTermId, int iSyncId, BOOL bInput)
{
    char *szForm;
    char *ptr;
    char szTmp[64];
    char szW_Zip[26];
    char szD_Zip[26];
    char szC_Zip[26];
    char szC_Phone[26];
    char szTmpStr1[122];
    char szTmpStr2[122];
    char szTmpStr3[122];
    char szTmpStr4[122];
    int i;
    int l;
    char *szZipPic = "XXXXX-XXXX";

    szForm = (char *)Term.pClientData[iTermId].szBuffer;

    Term.pClientData[iTermId].paymentData.w_id = Term.pClientData[iTermId].w_id;

    strcpy(szForm, " <HTML><HEAD><TITLE>TPC-C Payment</TITLE></HEAD><BODY>"
        "<FORM ACTION='tpcc.dll' METHOD='GET'>");

    if ( bInput )
        strcat(szForm, "<INPUT TYPE='hidden' NAME='PI#' VALUE='\"");

    strcat(szForm, "<INPUT TYPE='hidden' NAME='STATUSID' VALUE='\"");
    sprintf(szForm+strlen(szForm), "<INPUT TYPE='hidden' NAME='FORMID' VALUE='\"%d\">", PAYMENT_FORM);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE='hidden' NAME='TERMINID' VALUE='\"%d\">", iTermId);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE='hidden' NAME='SYNCID' VALUE='\"%d\">", iSyncId);
}
```

## Appendix A - Application Source Code

```
strcat(szForm, "<PRE>                Payment<BR>");

if ( bInput )
    strcat(szForm, "Date:<BR><BR>");
else
{
    wsprintf(szForm+strlen(szForm), "Date: %2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d <BR><BR>",
        Term.pClientData[iTermId].paymentData.h_date.day,
        Term.pClientData[iTermId].paymentData.h_date.month,
        Term.pClientData[iTermId].paymentData.h_date.year,
        Term.pClientData[iTermId].paymentData.h_date.hour,
        Term.pClientData[iTermId].paymentData.h_date.minute,
        Term.pClientData[iTermId].paymentData.h_date.second);
}
wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d", Term.pClientData[iTermId].paymentData.w_id);

if ( bInput )
{
    strcat(szForm, "                District: <INPUT NAME=\"DID*\" SIZE=1><BR><BR><BR><BR><BR>"
        "Customer: <INPUT NAME=\"CID*\" SIZE=4>"
        "Cust-Warehouse: <INPUT NAME=\"CW*\" SIZE=4> "
        "Cust-District: <INPUT NAME=\"CDI*\" SIZE=1><BR>"
        "Name:                <INPUT NAME=\"CLT*\" SIZE=16>                Since:<BR>"
        "                        Credit:<BR>"
        "                        Disc:<BR>"
        "                        Phone:<BR><B R>"
        "Amount Paid:    $<INPUT NAME=\"HAM*\" SIZE=7>    New Cust

Balance:<BR>"
        "Credit Limit:<BR><BR>Cust-Data: <BR><BR><BR><BR><PRE><HR>"
        "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\"><INPUT

TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">"
        "</BODY></FORM></HTML>");
}
else
{
    sprintf(szForm+strlen(szForm), "                District: %2.2d<BR>",
        Term.pClientData[iTermId].paymentData.d_id);

    FormatHTMLString(szTmpStr1, Term.pClientData[iTermId].paymentData.w_street_1, 20);
    FormatHTMLString(szTmpStr2, Term.pClientData[iTermId].paymentData.d_street_1, 20);

    sprintf(szForm+strlen(szForm), "%s                %s<BR>", szTmpStr1, szTmpStr2);

    FormatHTMLString(szTmpStr1, Term.pClientData[iTermId].paymentData.w_street_2, 20);
    FormatHTMLString(szTmpStr2, Term.pClientData[iTermId].paymentData.d_street_2, 20);

    sprintf(szForm+strlen(szForm), "%s                %s<BR>", szTmpStr1, szTmpStr2);

    FormatString(szW_Zip, szZipPic, Term.pClientData[iTermId].paymentData.w_zip);
    FormatString(szD_Zip, szZipPic, Term.pClientData[iTermId].paymentData.d_zip);

    FormatHTMLString(szTmpStr1, Term.pClientData[iTermId].paymentData.w_city, 20);
    FormatHTMLString(szTmpStr2, Term.pClientData[iTermId].paymentData.w_state, 2);
    FormatHTMLString(szTmpStr3, Term.pClientData[iTermId].paymentData.d_city, 20);
    FormatHTMLString(szTmpStr4, Term.pClientData[iTermId].paymentData.d_state, 2);

    wsprintf(szForm+strlen(szForm), "%s %s %10.10s    %s %s %10.10s<BR><BR>",
        szTmpStr1, szTmpStr2, szW_Zip, szTmpStr3, szTmpStr4, szD_Zip );

    wsprintf(szForm+strlen(szForm), "Customer: %4.4d Cust-Warehouse: %4.4d Cust-District: %2.2d<BR>",
        Term.pClientData[iTermId].paymentData.c_id,
        Term.pClientData[iTermId].paymentData.c_w_id,
        Term.pClientData[iTermId].paymentData.c_d_id);

    FormatHTMLString(szTmpStr1, Term.pClientData[iTermId].paymentData.c_first, 16);
    FormatHTMLString(szTmpStr2, Term.pClientData[iTermId].paymentData.c_middle, 2);
    FormatHTMLString(szTmpStr3, Term.pClientData[iTermId].paymentData.c_last, 16);

    wsprintf(szForm+strlen(szForm), "Name: %s %s %s    Since: %2.2d-%2.2d-%4.4d<BR>",
        szTmpStr1, szTmpStr2, szTmpStr3,
        Term.pClientData[iTermId].paymentData.c_since.day,
        Term.pClientData[iTermId].paymentData.c_since.month,
        Term.pClientData[iTermId].paymentData.c_since.year);

    FormatHTMLString(szTmpStr1, Term.pClientData[iTermId].paymentData.c_street_1, 20);
    FormatHTMLString(szTmpStr2, Term.pClientData[iTermId].paymentData.c_credit, 2);

    wsprintf(szForm+strlen(szForm), "                %s                Credit: %s<BR>", szTmpStr1, szTmpStr2);

    FormatHTMLString(szTmpStr1, Term.pClientData[iTermId].paymentData.d_street_2, 20);
    sprintf(szForm+strlen(szForm), "                %s                %s Disc: %5.2f<BR>",
        szTmpStr1, Term.pClientData[iTermId].paymentData.c_discount);

    FormatString(szC_Zip, szZipPic, Term.pClientData[iTermId].paymentData.c_zip);
    FormatString(szC_Phone, "XXXXXX-XXX-XXX-XXXX", Term.pClientData[iTermId].paymentData.c_phone);

    FormatHTMLString(szTmpStr1, Term.pClientData[iTermId].paymentData.c_city, 20);
    FormatHTMLString(szTmpStr2, Term.pClientData[iTermId].paymentData.c_state, 2);

    wsprintf(szForm+strlen(szForm), "                %s %s %10.10s    Phone: %-19.19s<BR><BR>",
```

## Appendix A - Application Source Code

```
szTmpStr1, szTmpStr2, szC_Zip, szC_Phone );

sprintf(szForm+strlen(szForm), "Amount Paid:   $%7.2f   New Cust Balance: $%14.2f<BR>",
Term.pClientData[iTermId].paymentData.h_amount,
Term.pClientData[iTermId].paymentData.c_balance);

sprintf(szForm+strlen(szForm), "Credit Limit:  $%13.2f<BR><BR>",
Term.pClientData[iTermId].paymentData.c_credit_lim);

ptr = Term.pClientData[iTermId].paymentData.c_credit;
if ( *ptr == 'B' && *(ptr+1) == 'C' )
{
    ptr = Term.pClientData[iTermId].paymentData.c_data;
    l = strlen( ptr ) / 50;
    for(i=0; i<4; i++, ptr += 50)
    {
        if ( i <= 1)
            UtilStrCpy(szTmp, ptr, 50);
        else
            szTmp[0] = 0;
        if ( !i )
        {
            FormatHTMLString(szTmpStr1, szTmp, 50);
            sprintf(szForm+strlen(szForm), "Cust-Data: %s<BR>", szTmpStr1);
        }
        else
        {
            FormatHTMLString(szTmpStr1, szTmp, 50);
            sprintf(szForm+strlen(szForm), "    %s<BR>", szTmpStr1);
        }
    }
}
else
    strcat(szForm, "Cust-Data: <BR><BR><BR><BR>");

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

                                "<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>");

}

return szForm;
}

/* FUNCTION: char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL bInput)
 *
 * PURPOSE: This function
 *
 * ARGUMENTS:      int                iTermId      client browser terminal id
 *                  int                iSyncId      client browser sync id
 *                  BOOL               bInput       TRUE if form is being constructed for input else FALSE
 *
 * RETURNS:        char *              A pointer to buffer inside client structure where HTML form is built.
 *
 * 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.
 */

static char *MakeOrderStatusForm(int iTermId, int iSyncId, BOOL bInput)
{
    char      *szForm;
    char      c_first[98];
    char      c_middle[14];
    char      c_last[98];
    int       i;

    szForm = (char *)Term.pClientData[iTermId].szBuffer;

    Term.pClientData[iTermId].orderStatusData.w_id = Term.pClientData[iTermId].w_id;

    strcpy(szForm, " <HTML><HEAD><TITLE>TPC-C Order-Status</TITLE></HEAD><BODY>"
                  " <FORM ACTION=\"/tpcc.dll\" METHOD=\"GET\">");

    if ( bInput )
        strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"PI#\" VALUE=\"\">");

    strcat(szForm, "<INPUT TYPE=\"hidden\" NAME=\"STATUSID\" VALUE=\"0\">");
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"FORMID\" VALUE=\"%d\">", ORDER_STATUS_FORM);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"TERMINID\" VALUE=\"%d\">", iTermId);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE=\"hidden\" NAME=\"SYCNID\" VALUE=\"%d\">", iSyncId);

    strcat(szForm, "<PRE>                Order-Status<BR>");
    sprintf(szForm+strlen(szForm), "Warehouse: %4.4d   ", Term.pClientData[iTermId].orderStatusData.w_id);

    if ( bInput )
    {
        strcat(szForm, "District: <INPUT NAME=\"DID#\" SIZE=1><BR>"

```

# Appendix A - Application Source Code

```
NAME="CLT*" SIZE=23><BR>"
"Customer: <INPUT NAME="CID*" SIZE=4> Name: <INPUT
" Cust-Balance:<BR><BR>"
"Order-Number: Entry-Date: Carrier-Number:<BR>"
"Supply-W Item-Id Qty Amount Delivery-Date<BR></PRE>"
" <HR><INPUT TYPE="submit" NAME="CMD" VALUE="Process"><INPUT
" </BODY></FORM></HTML> );
}
else
{
    sprintf(szForm+strlen(szForm), "District: %2.2d<BR>", Term.pClientData[iTermId].orderStatusData.d_id);

    FormatHTMLString(c_first, Term.pClientData[iTermId].orderStatusData.c_first, 16);
    FormatHTMLString(c_middle, Term.pClientData[iTermId].orderStatusData.c_middle, 2);
    FormatHTMLString(c_last, Term.pClientData[iTermId].orderStatusData.c_last, 16);

    sprintf(szForm+strlen(szForm), "Customer: %4.4d Name: %s %s %s<BR>",
        Term.pClientData[iTermId].orderStatusData.c_id, c_first, c_middle, c_last);

    sprintf(szForm+strlen(szForm), "Cust-Balance: $%9.2f<BR><BR>",
        Term.pClientData[iTermId].orderStatusData.c_balance);

    sprintf(szForm+strlen(szForm), "Order-Number: %8.8d Entry-Date: %2.2d-%2.2d-%4.4d %2.2d:%2.2d:%2.2d Carrier-Number: %2.2d<BR>",
        Term.pClientData[iTermId].orderStatusData.o_id,
        Term.pClientData[iTermId].orderStatusData.o_entry_d.day,
        Term.pClientData[iTermId].orderStatusData.o_entry_d.month,
        Term.pClientData[iTermId].orderStatusData.o_entry_d.year,
        Term.pClientData[iTermId].orderStatusData.o_entry_d.hour,
        Term.pClientData[iTermId].orderStatusData.o_entry_d.minute,
        Term.pClientData[iTermId].orderStatusData.o_entry_d.second,
        Term.pClientData[iTermId].orderStatusData.o_carrier_id);

    strcat(szForm+strlen(szForm), "Supply-W Item-Id Qty Amount Delivery-Date<BR>");

    for(i=0; i<Term.pClientData[iTermId].orderStatusData.o_ol_cnt; i++)
    {
        sprintf(szForm+strlen(szForm), " %4.4d %6.6d %2.2d $%8.2f %2.2d-%2.2d-%4.4d<BR>",
            Term.pClientData[iTermId].orderStatusData.OIOrderStatusData[i].ol_supply_w_id,
            Term.pClientData[iTermId].orderStatusData.OIOrderStatusData[i].ol_i_id,
            Term.pClientData[iTermId].orderStatusData.OIOrderStatusData[i].ol_quantity,
            Term.pClientData[iTermId].orderStatusData.OIOrderStatusData[i].ol_amount,
            Term.pClientData[iTermId].orderStatusData.OIOrderStatusData[i].ol_delivery_d.day,
            Term.pClientData[iTermId].orderStatusData.OIOrderStatusData[i].ol_delivery_d.month,
            Term.pClientData[iTermId].orderStatusData.OIOrderStatusData[i].ol_delivery_d.year);
    }

    strcat(szForm, "<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..">"
        "</BODY></FORM></HTML> );
}

return szForm;
}

/* FUNCTION: char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput)
*
* PURPOSE: This function
*
* ARGUMENTS: int iTermId client browser terminal id
* int iSyncId client browser sync id
* BOOL bInput TRUE if form is being constructed for input else FALSE
*
* RETURNS: char * A pointer to buffer inside client structure where HTML form is built.
*
* 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.
*/

static char *MakeDeliveryForm(int iTermId, int iSyncId, BOOL bInput)
{
    char *szForm;

    szForm = (char *)Term.pClientData[iTermId].szBuffer;

    Term.pClientData[iTermId].deliveryData.w_id = Term.pClientData[iTermId].w_id;

    strcpy( szForm, " <HTML><HEAD><TITLE>TPC-C Delivery</TITLE></HEAD><BODY>"
        "<FORM ACTION="tpcc.dll" METHOD="GET">");

    if ( bInput )
        strcat(szForm, "<INPUT TYPE="hidden" NAME="PI*" VALUE=" ">");

    strcat(szForm, "<INPUT TYPE="hidden" NAME="STATUSID" VALUE="0">");
    sprintf(szForm+strlen(szForm), "<INPUT TYPE="hidden" NAME="FORMID" VALUE="%d">, DELIVERY_FORM);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE="hidden" NAME="TERMINID" VALUE="%d">", iTermId);
    sprintf(szForm+strlen(szForm), "<INPUT TYPE="hidden" NAME="SYNCID" VALUE="%d">", iSyncId);
}
```

## Appendix A - Application Source Code

```
strcat(szForm, "<PRE>                Delivery<BR>");

wprintf(szForm+strlen(szForm), "Warehouse: %4.4d<BR><BR>", Term.pClientData[iTermId].deliveryData.w_id);

if ( bInput )
    strcat( szForm, "Carrier Number: <INPUT NAME=\"OCD\" SIZE=1><BR><BR>");
else
{
    wprintf(szForm+strlen(szForm), "Carrier Number: %2.2d<BR><BR>",
        Term.pClientData[iTermId].deliveryData.o_carrier_id);
}
if ( bInput )
{
    strcat( szForm, "Execution Status:<BR><PRE>"
        "<HR><INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Process\">"
        "<INPUT TYPE=\"submit\" NAME=\"CMD\" VALUE=\"Menu\">");
}
else
{
    wprintf(szForm+strlen(szForm), "Execution Status: %25.25s<BR><PRE>",
        Term.pClientData[iTermId].deliveryData.execution_status);

    strcat(szForm, "<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..\">");
}

strcat( szForm,
        "</BODY></FORM></HTML>");

return szForm;
}

/* 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.
 */

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

    return;
}

/* FUNCTION: void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
 * 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.
 * ARGUMENTS:      EXTENSION_CONTROL_BLOCK *pECB    passed in structure pointer from inetrv.
 *                  int iTermId                    client browser
 *                  int iSyncId                    iSyncId client browser sync id
 * RETURNS:        None
 * COMMENTS:       None
 */

static void ProcessNewOrderForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
{
    int iRc;
    int iError;
    PECBINFO pEcbInfo;

    memset(&Term.pClientData[iTermId].newOrderData, 0, sizeof(NEW_ORDER_DATA));

    Term.pClientData[iTermId].newOrderData.w_id = Term.pClientData[iTermId].w_id;

    if ( (iError=GetNewOrderData(pECB->lpszQueryString, &Term.pClientData[iTermId].newOrderData)) != ERR_SUCCESS )
    {
        ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }
}
```

## Appendix A - Application Source Code

---

```
    }

    iRc = SQLNewOrder(pECB, iTermId, iSyncId, Term.pClientData[iTermId].dbproc, &Term.pClientData[iTermId].newOrderData, iDeadlockRetry);

    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(Term.pClientData[iTermId].dbproc)) )
    {
        if ( pEcbInfo->bFailed )
            return;
    }

    if ( iRc < 0 )
        ErrorMessage(pECB, ERR_NEW_ORDER_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    else
        WriteZString(pECB, MakeNewOrderForm(iTermId, iSyncId, FALSE, (BOOL)iRc) );

    return;
}

/* FUNCTION: void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
 * 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
 *                 int iSyncId                       iSyncId client browser sync id
 * RETURNS:        None
 * COMMENTS:      None
 */

static void ProcessPaymentForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
{
    int iRc;
    int iError;
    PECBINFO pEcbInfo;

    memset(&Term.pClientData[iTermId].paymentData, 0, sizeof(PAYMENT_DATA));

    Term.pClientData[iTermId].paymentData.w_id = Term.pClientData[iTermId].w_id;

    if ( (iError=GetPaymentData(pECB->lpszQueryString, &Term.pClientData[iTermId].paymentData) != ERR_SUCCESS ) )
    {
        ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    iRc = SQLPayment(pECB, iTermId, iSyncId, Term.pClientData[iTermId].dbproc, &Term.pClientData[iTermId].paymentData, iDeadlockRetry);

    if ( (pEcbInfo = (PECBINFO)dbgetuserdata(Term.pClientData[iTermId].dbproc)) )
    {
        if ( pEcbInfo->bFailed )
            return;
    }

    if ( iRc == 0 )
        ErrorMessage(pECB, ERR_PAYMENT_INVALID_CUSTOMER, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    else if ( iRc < 0 )
        ErrorMessage(pECB, ERR_PAYMENT_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    else
        WriteZString(pECB, MakePaymentForm(iTermId, iSyncId, FALSE) );

    return;
}

/* FUNCTION: void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
 * 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
 *                 int iSyncId                       iSyncId client browser sync id
 * RETURNS:        None
 * COMMENTS:      None
 */

static void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
{
    int iRc;
```

## Appendix A - Application Source Code

---

```
int iError;
PECBINFO pEcbInfo;

memset(&Term.pClientData[iTermId].orderStatusData, 0, sizeof(ORDER_STATUS_DATA));
Term.pClientData[iTermId].orderStatusData.w_id = Term.pClientData[iTermId].w_id;

if ( (iError=GetOrderStatusData(pECB->lpszQueryString, &Term.pClientData[iTermId].orderStatusData)) != ERR_SUCCESS )
{
    ErrorMessage(pECB, iError, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    return;
}

iRc = SQLOrderStatus(pECB, iTermId, iSyncId, Term.pClientData[iTermId].dbproc, &Term.pClientData[iTermId].orderStatusData, iDeadlockRetry);
if ( (pEcbInfo = (PECBINFO)dbgetuserdata(Term.pClientData[iTermId].dbproc)) )
{
    if ( pEcbInfo->bFailed )
        return;
}

if ( iRc == 0 )
    ErrorMessage(pECB, ERR_NOSUCH_CUSTOMER, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
else if ( iRc < 0 )
    ErrorMessage(pECB, ERR_ORDER_STATUS_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
else
    WriteZString(pECB, MakeOrderStatusForm(iTermId, iSyncId, FALSE) );

return;
}

/* FUNCTION: void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
* 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 inetrv.
* terminal id int iTermId client browser
* sync id int iSyncId client browser
* RETURNS: None
* COMMENTS: None
*/

static void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
{
    char szTmp[26];

    memset(&Term.pClientData[iTermId].ddiveryData, 0, sizeof(DELIVERY_DATA));
    Term.pClientData[iTermId].deliveryData.w_id = Term.pClientData[iTermId].w_id;

    if ( !GetKeyValue(pECB->lpszQueryString, "OCD*", szTmp, sizeof(szTmp)) )
    {
        ErrorMessage(pECB, ERR_DELIVERY_MISSING_OCD_KEY, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    if ( !IsNumeric(szTmp) )
    {
        ErrorMessage(pECB, ERR_DELIVERY_CARRIER_INVALID, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    Term.pClientData[iTermId].deliveryData.o_carrier_id = atoi(szTmp);

    if ( Term.pClientData[iTermId].deliveryData.o_carrier_id > 10 || Term.pClientData[iTermId].deliveryData.o_carrier_id < 1 )
    {
        ErrorMessage(pECB, ERR_DELIVERY_CARRIER_ID_RANGE, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    //post delivery info
    if ( PostDeliveryInfo(Term.pClientData[iTermId].deliveryData.w_id, Term.pClientData[iTermId].deliveryData.o_carrier_id) )
        strcpy(Term.pClientData[iTermId].deliveryData.execution_status, "Delivery Post Failed");
    else
        strcpy(Term.pClientData[iTermId].ddiveryData.execution_status, "Delivery has been queued.");

    WriteZString(pECB, MakeDeliveryForm(iTermId, iSyncId, FALSE) );

    return;
}

/* FUNCTION: void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
* PURPOSE: This function gets and validates the input data from the Stock Level
```

## Appendix A - Application Source Code

---

```
*
*                               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.
* terminal id
*                               int                               iTermId      client browser
*                               int                               iSyncId      client browser
* sync id
*
* RETURNS:        None
*
* COMMENTS:       None
*
*/

static void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId)
{
    char          szTmp[26];
    BOOL          bRc;
    PECBINFO      pEcbInfo;

    memset(&Term.pClientData[iTermId].stockLevelData, 0, sizeof(STOCK_LEVEL_DATA));

    Term.pClientData[iTermId].stockLevelData.w_id = Term.pClientData[iTermId].w_id;
    Term.pClientData[iTermId].stockLevelData.d_id = Term.pClientData[iTermId].d_id;

    if ( !GetKeyValue(pECB->lpszQueryString, "TT*", szTmp, sizeof(szTmp)) )
    {
        ErrorMessage(pECB, ERR_STOCKLEVEL_MISSING_THRESHOLD_KEY, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    if ( !IsNumeric(szTmp) )
    {
        ErrorMessage(pECB, ERR_STOCKLEVEL_THRESHOLD_INVALID, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    Term.pClientData[iTermId].stockLevelData.thresh_hold = atoi(szTmp);

    if ( Term.pClientData[iTermId].stockLevelData.thresh_hold >= 100 || Term.pClientData[iTermId].stockLevelData.thresh_hold < 0 )
    {
        ErrorMessage(pECB, ERR_STOCKLEVEL_THRESHOLD_RANGE, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
        return;
    }

    bRc = SQLStockLevel(pECB, iTermId, iSyncId, Term.pClientData[iTermId].dbproc, &Term.pClientData[iTermId].stockLevelData, iDeadlockRetry);

    if ( pEcbInfo = (PECBINFO)dbgetuserdata(Term.pClientData[iTermId].dbproc) )
    {
        if ( pEcbInfo->bFailed )
            return;
    }

    if ( bRc )
        ErrorMessage(pECB, ERR_STOCKLEVEL_NOT_PROCESSED, ERR_TYPE_WEBDLL, NULL, iTermId, iSyncId);
    else
        WriteZString(pECB, MakeStockLevelForm(iTermId, iSyncId, FALSE) );

    return;
}

/* FUNCTION: int GetNewOrderData(LPSTR lpszQueryString, NEW_ORDER_DATA *pNewOrderData)
*
* 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
*
* RETURNS:        int                               error code
* indicating reason for failure
*                               ERR_SUCCESS
*                               new order input data successfully parsed
*
* COMMENTS:       None
*
*/

static int GetNewOrderData(LPSTR lpszQueryString, NEW_ORDER_DATA *pNewOrderData)
{
    char          szTmp[26];
    char          szKey[26];
    int           i;
    short         items;
    BOOL          bCheck;

    if ( !GetKeyValue(lpszQueryString, "DID*", szTmp, sizeof(szTmp)) )
        return ERR_NEWORDER_FORM_MISSING_DID;
}
```



## Appendix A - Application Source Code

---

```
if ( !IsNumeric(szTmp) )
    return ERR_NEWORDER_DISTRICT_INVALID;

pNewOrderData->d_id = atoi(szTmp);

if ( !GetKeyValue(lpszQueryString, "CID#", szTmp, sizeof(szTmp)) )
    return ERR_NEWORDER_CUSTOMER_KEY;

if ( !IsNumeric(szTmp) )
    return ERR_NEWORDER_CUSTOMER_INVALID;
pNewOrderData->c_id = atoi(szTmp);

bCheck = FALSE;
for(i=0, items=0; i<15; i++)
{
    wsprintf(szKey, "IID%2.2d*", i);
    if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
        return ERR_NEWORDER_MISSING_IID_KEY;
    if ( szTmp[0] )
    {
        //if blank lines between item ids
        if ( bCheck )
            return ERR_NEWORDER_ITEM_BLANK_LINES;
        if ( !IsNumeric(szTmp) )
            return ERR_NEWORDER_ITEMID_INVALID;
        pNewOrderData->Ol[i].ol_i_id = atoi(szTmp);

        wsprintf(szKey, "SP%2.2d*", i);
        if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
            return ERR_NEWORDER_MISSING_SUPPW_KEY;
        if ( !IsNumeric(szTmp) )
            return ERR_NEWORDER_SUPPW_INVALID;
        pNewOrderData->Ol[i].ol_supply_w_id = (short)atoi(szTmp);

        wsprintf(szKey, "Qty%2.2d*", i);
        if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
            return ERR_NEWORDER_MISSING_QTY_KEY;

        if ( !IsNumeric(szTmp) )
            return ERR_NEWORDER_QTY_INVALID;

        pNewOrderData->Ol[i].ol_quantity = atoi(szTmp);
        items++;

        if ( pNewOrderData->Ol[i].ol_i_id >= 1000000 || pNewOrderData->Ol[i].ol_i_id < 1 )
            return ERR_NEWORDER_ITEMID_RANGE;
        if ( pNewOrderData->Ol[i].ol_quantity >= 100 || pNewOrderData->Ol[i].ol_quantity < 1 )
            return ERR_NEWORDER_QTY_RANGE;
    }
    else
    {
        wsprintf(szKey, "SP%2.2d*", i);
        if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
            return ERR_NEWORDER_MISSING_QTY_KEY;

        if ( szTmp[0] )
            return ERR_NEWORDER_SUPPW_WITHOUT_ITEMID;

        wsprintf(szKey, "Qty%2.2d*", i);
        if ( !GetKeyValue(lpszQueryString, szKey, szTmp, sizeof(szTmp)) )
            return ERR_NEWORDER_MISSING_QTY_KEY;

        if ( szTmp[0] )
            return ERR_NEWORDER_QTY_WITHOUT_ITEMID;

        bCheck = TRUE;
    }
}
if ( items == 0 )
    return ERR_NEWORDER_NOITEMS_ENTERED;

pNewOrderData->o_ol_cnt = items;

return ERR_SUCCESS;
}

/* FUNCTION: int GetPaymentData(LPSTR lpszQueryString, PAYMENT_DATA *pPaymentData)
 *
 * 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
 *
 * RETURNS:        int                    error code
 *                  indicating reason for failure
 *
 *                  ERR_SUCCESS
 *                  all input data successfully parsed
 *
 * COMMENTS:      None
 *
 */
```

## Appendix A - Application Source Code

---

```
static int GetPaymentData(LPSTR lpszQueryString, PAYMENT_DATA *pPaymentData)
{
    char        szTmp[26];
    char        *ptr;

    if ( !GetKeyValue(lpszQueryString, "DID*", szTmp, sizeof(szTmp)) )
        return ERR_PAYMENT_MISSING_DID_KEY;
    if ( !IsNumeric(szTmp) )
        return ERR_PAYMENT_DISTRICT_INVALID;
    pPaymentData->d_id = atoi(szTmp);

    if ( !GetKeyValue(lpszQueryString, "CID*", szTmp, sizeof(szTmp)) )
        return ERR_PAYMENT_MISSING_CID_KEY;

    if ( szTmp[0] && !IsNumeric(szTmp) )
        return ERR_PAYMENT_CUSTOMER_INVALID;

    pPaymentData->c_id = atoi(szTmp);

    if ( szTmp[0] == 0 )
    {
        if ( !GetKeyValue(lpszQueryString, "CLT*", szTmp, sizeof(szTmp)) )
            return ERR_PAYMENT_MISSING_CLT;
        _strupr( szTmp );

        strcpy(pPaymentData->c_last, szTmp);
        if ( strlen(pPaymentData->c_last) > 16 )
            return ERR_PAYMENT_LAST_NAME_TO_LONG;
    }
    else
    {
        if ( !GetKeyValue(lpszQueryString, "CLT*", szTmp, sizeof(szTmp)) )
            return ERR_PAYMENT_MISSING_CLT_KEY;
        if ( szTmp[0] )
            return ERR_PAYMENT_CID_AND_CLT;
    }

    if ( !GetKeyValue(lpszQueryString, "CDI*", szTmp, sizeof(szTmp)) )
        return ERR_PAYMENT_MISSING_CDI_KEY;
    if ( !IsNumeric(szTmp) )
        return ERR_PAYMENT_CDI_INVALID;
    pPaymentData->c_d_id = atoi(szTmp);

    if ( !GetKeyValue(lpszQueryString, "CWI*", szTmp, sizeof(szTmp)) )
        return ERR_PAYMENT_MISSING_CWI_KEY;

    if ( !IsNumeric(szTmp) )
        return ERR_PAYMENT_CWI_INVALID;

    pPaymentData->c_w_id = atoi(szTmp);

    if ( !GetKeyValue(lpszQueryString, "HAM*", szTmp, sizeof(szTmp)) )
        return ERR_PAYMENT_MISSING_HAM_KEY;

    ptr = szTmp;
    while( *ptr )
    {
        if ( *ptr == ',' )
        {
            ptr++;
            if ( !*ptr )
                break;
            if ( *ptr < '0' || *ptr > '9' )
                return ERR_PAYMENT_HAM_INVALID;

            ptr++;
            if ( !*ptr )
                break;
            if ( *ptr < '0' || *ptr > '9' )
                return ERR_PAYMENT_HAM_INVALID;

            if ( !*ptr )
                return ERR_PAYMENT_HAM_INVALID;
        }
        else if ( *ptr < '0' || *ptr > '9' )
            return ERR_PAYMENT_HAM_INVALID;

        ptr++;
    }

    pPaymentData->h_amount = atof(szTmp);
    if ( pPaymentData->h_amount >= 10000.00 || pPaymentData->h_amount < 0 )
        return ERR_PAYMENT_HAM_RANGE;

    return ERR_SUCCESS;
}

/* FUNCTION: int GetOrderStatusData(LPSTR lpszQueryString, ORDER_STATUS_DATA *pOrderStatusData)
 *
 * PURPOSE: This function extracts and validates the payment form data from an http command string.
```

## Appendix A - Application Source Code

---

```
*
* ARGUMENTS:      LPSTR                IpszQueryString                client browser http command string
*                  ORDER_STATUS_DATA  *pOrderStatusData            pointer to order status data structure
*
* RETURNS:        int                  error code
* indicating reason for failure
*                  ERR_SUCCESS
*                  successfully parsed all required input data
*
* COMMENTS:       None
*
*/
static int GetOrderStatusData(LPSTR IpszQueryString, ORDER_STATUS_DATA *pOrderStatusData)
{
    char        szTmp[26];

    if ( !GetKeyValue(IpszQueryString, "DID*", szTmp, sizeof(szTmp)) )
        return ERR_ORDERSTATUS_MISSING_DID_KEY;
    if ( !IsNumeric(szTmp) )
        return ERR_ORDERSTATUS_DID_INVALID;
    pOrderStatusData->d_id = atoi(szTmp);

    if ( !GetKeyValue(IpszQueryString, "CID*", szTmp, sizeof(szTmp)) )
        return ERR_ORDERSTATUS_MISSING_CID_KEY;

    if ( szTmp[0] == 0 )
    {
        pOrderStatusData->c_id = 0;
        if ( !GetKeyValue(IpszQueryString, "CLT*", szTmp, sizeof(szTmp)) )
            return ERR_ORDERSTATUS_MISSING_CLT_KEY;
        _strupr( szTmp );
        strcpy(pOrderStatusData->c_last, szTmp);
        if ( strlen(pOrderStatusData->c_last) > 16 )
            return ERR_ORDERSTATUS_CLT_RANGE;
    }
    else
    {
        if ( !IsNumeric(szTmp) )
            return ERR_ORDERSTATUS_CID_INVALID;
        pOrderStatusData->c_id = atoi(szTmp);
        if ( !GetKeyValue(IpszQueryString, "CLT*", szTmp, sizeof(szTmp)) )
            return ERR_ORDERSTATUS_MISSING_CLT_KEY;
        if ( szTmp[0] )
            return ERR_ORDERSTATUS_CID_AND_CLT;
    }

    return ERR_SUCCESS;
}

/* FUNCTION: BOOL ReadRegistrySettings(void)
*
* PURPOSE: This function reads the NT registry for startup parameters. There parameters are
*           under the TPCC key.
*
* ARGUMENTS: None
*
* RETURNS:   None
*
* COMMENTS:  This function also sets up required operation variables to their default value
*             so if registry is not setup the default values will be used.
*/
static BOOL ReadRegistrySettings(void)
{
    HKEY    hKey;
    DWORD   size;
    DWORD   type;
    char    szTmp[256];

    bLog                = FALSE;
    iMaxWareHouses      = 500;
    iThreads             = 5;
    iDelayMs             = 100;
    iDeadlockRetry       = (short)3;
    strcpy(szTpccLogPath, "tpcclog.");

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

    if ( RegQueryValueEx(hKey, "PATH", 0, &type, szTmp, &size) == ERROR_SUCCESS )
    {
        strcpy(szTpccLogPath, szTmp);
        strcat(szTpccLogPath, "tpcclog.");
        strcpy(szErrorLogPath, szTmp);
        strcat(szErrorLogPath, "tpccerr.");
    }

    size = sizeof(szTmp);
}
```

## Appendix A - Application Source Code

---

```
if ( RegQueryValueEx(hKey, "LOG", 0, &type, szTmp, &size) == ERROR_SUCCESS )
{
    if ( !strcmp(szTmp, "ON") )
        bLog = TRUE;
}

size = sizeof(szTmp);
if ( RegQueryValueEx(hKey, "MaximumWarehouses", 0, &type, szTmp, &size) == ERROR_SUCCESS )
{
    iMaxWareHouses = atoi(szTmp);
    if ( iMaxWareHouses == 0 )
        iMaxWareHouses = 500;
}

size = sizeof(szTmp);
if ( RegQueryValueEx(hKey, "NumberOfDeliveryThreads", 0, &type, szTmp, &size) == ERROR_SUCCESS )
    iThreads = atoi(szTmp);
if ( !iThreads )
    iThreads = 5;

size = sizeof(szTmp);
if ( RegQueryValueEx(hKey, "BackoffDelay", 0, &type, szTmp, &size) == ERROR_SUCCESS )
    iDelayMs = atoi(szTmp);
if ( !iDelayMs )
    iDelayMs = 100;

size = sizeof(szTmp);
if ( RegQueryValueEx(hKey, "DeadlockRetry", 0, &type, szTmp, &size) == ERROR_SUCCESS )
    iDeadlockRetry = (short)atoi(szTmp);
if ( !iDeadlockRetry )
    iDeadlockRetry = (short)3;

size = sizeof(szTmp);
if ( RegQueryValueEx(hKey, "MaxConnections", 0, &type, szTmp, &size) == ERROR_SUCCESS )
    iMaxConnections = (short)atoi(szTmp);
if ( !iMaxConnections )
    iMaxConnections = (short)25;

RegCloseKey(hKey);

return FALSE;
}

/* FUNCTION: BOOL PostDeliveryInfo(short w_id, short o_carrier_id)
 *
 * PURPOSE: This function writes the delivery information to the delivery pipe. The information is
 *          sent as a long.
 *
 * ARGUMENTS:      short          short          warehouse id
 *                  short          o_carrier_id  carrier id
 *
 * RETURNS:        BOOL          FALSE          delivery information posted successfully
 *                  TRUE          error cannot post delivery info
 *
 * COMMENTS:       The pipe is initially created with 16K buffer size this should allow for
 *                  up to 4096 deliveries to be queued before an overflow condition would
 *                  occur. The only reason that an overflow would occur is if the delivery
 *                  application stopped listening while deliveries were being posted.
 */

static BOOL PostDeliveryInfo(short w_id, short o_carrier_id)
{
    DELIVERY_TRANSACTION    deliveryTransaction;
    int                     d;
    int                     i;

    GetLocalTime(&deliveryTransaction.queue);

    deliveryTransaction.w_id      = w_id;
    deliveryTransaction.o_carrier_id = o_carrier_id;

    for(i=0; i<4; i++)
    {
        if ( WriteFile(hPipe, &deliveryTransaction, sizeof(deliveryTransaction), &d, NULL) )
            return FALSE;

        if ( GetLastError() != ERROR_PIPE_BUSY ) //ERROR_PIPE_LISTENING
            return TRUE;
    }

    return TRUE;
}

/* 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.
```

## Appendix A - Application Source Code

---

```
*
* RETURNS:          BOOL      FALSE   if string is not all numeric      TRUE   if string contains only numeric characters i.e. '0' - '9'
*
* COMMENTS:        None
*/

static BOOL IsNumeric(char *ptr)
{
    if ( *ptr == 0 )
        return FALSE;

    while( *ptr && isdigit(*ptr) )
        ptr++;
    return ( !*ptr );
}

#ifdef USE_ODBC
/* FUNCTION: static int ODBCError(DBPROCESS *dbproc)
*
* PURPOSE: This function Handles the processing of errors from ODBC APIs
*
* ARGUMENTS:      PDBPROCESS
*
* RETURNS:        BOOL      FALSE   if string is not all numeric      TRUE   if string contains only numeric characters i.e. '0' - '9'
*
* COMMENTS:      None
*/

static int ODBCError(DBPROCESS *dbproc)
{
    RETCODE   rc;
    SDWORD   lNativeError;
    BOOL      bError;
    char      szState[6];
    char      szMsg[SQL_MAX_MESSAGE_LENGTH];
    char      timebuf[128];
    char      datebuf[128];

    dbproc->deadlock_detected = TRUE;
    bError = FALSE;

    while( SQLError(dbproc->henv,

                    dbproc->hdbc,
                    dbproc->hstmt,
                    szState,
                    &lNativeError,
                    szMsg,
                    sizeof(szMsg),
                    NULL) != SQL_NO_DATA_FOUND )
    {
        if (lNativeError == 1205)
            dbproc->deadlock_detected = TRUE;
        else
        {
            _strtime(timebuf);
            _strdate(datebuf);

            h_printf(dbproc->pECB, "%s %s : ODBC Error: State=%s, Error=%ld, %s\n", datebuf, timebuf, szState, lNativeError,
szMsg);
            bError = TRUE;
        }
    }
    if ( bError )
        return -1;
    return dbproc->deadlock_detected;
}

#endif

/* FUNCTION: void FormatHTMLString(char *szBuff, int iLen, char *szStr)
*
* PURPOSE: This function Handles translation of HTML specific character field data
*          when an HTML output form is generated.
*
* ARGUMENTS:  char      *szBuff   Returned string information
*            char      *szStr   input string to be formatted.
*            int       iLen     Length of returned string
*
* RETURNS:    none
*
* COMMENTS:   The length paramter is the absolute length of the returned string in
*            HTML characters. For example the input string > would be returned as
*            &gt; which would be counted as 1 character.If the number of input
*            characters is less than the iLen parameter spaces are appended to
*            the end of the string to ensure that at least iLen characters are
*            returned in the szBuff parameter.
*/
```

## Appendix A - Application Source Code

---

```
static void FormatHTMLString(char *szBuff, char *szStr, int iLen)
{
    while( iLen && *szStr )
    {
        switch( *szStr )
        {
            case '>':
                *szBuff++ = '&';
                *szBuff++ = 'g';
                *szBuff++ = 't';
                *szBuff++ = ';';
                szStr++;
                break;

            case '<':
                *szBuff++ = '&';
                *szBuff++ = 'l';
                *szBuff++ = 't';
                *szBuff++ = ';';
                szStr++;
                break;

            case '&':
                *szBuff++ = '&';
                *szBuff++ = 'a';
                *szBuff++ = 'm';
                *szBuff++ = 'p';
                *szBuff++ = ';';
                szStr++;
                break;

            case '\\':
                *szBuff++ = '&';
                *szBuff++ = 'q';
                *szBuff++ = 'u';
                *szBuff++ = 'o';
                *szBuff++ = 't';
                *szBuff++ = ';';
                szStr++;
                break;

            default:
                *szBuff++ = *szStr++;
                break;
        }
        iLen--;
    }
    while( iLen-- )
        *szBuff++ = ' ';

    *szBuff = 0;

    return;
}
```

### WebcInt.c

```
/*      FILE:          WEBCNCT.C
 *
 *                      Microsoft TPC-C Kit Ver. 3.00.000
 *
 *                      Copyright Microsoft, 1996
 *
 *      PURPOSE:      Example application that connects a specified number of clients to
 *                    the WEB client TPCC.DLL.
 *
 *      REQUIREMENTS: Requires VC++ Version 4.2 for the wininet internet library.
 *
 *      Author:       Philip Durr
 *                    philipdu@Microsoft.com
 */

#include <windows.h>
#include <stdio.h>
#include <stdlib.h>
#include <memory.h>
#include <malloc.h>
#include <conio.h>
#include <wininet.h>

char szCmd1[256];
char szCmd2[256];
char szCmd3[256];

int versionMS = 3;
int versionMM = 0;
int versionLS = 2;

int          pakt(void);
void          Login(HINTERNET hInternet, int *pTermId, int *pSyncId);
int          Logout(HINTERNET hInternet, int iTermId, int iSyncId);
```

## Appendix A - Application Source Code

---

```
int ReadUrlPage(HINTERNET hInternet, char *szUrl, char **ppBuffer, int *piTotalBytes);
static BOOL GetParameters(int argc, char *argv[]);
static void PrintParameters(void);
static void PrintHeader(void);

int iTotalConnections;
int iTermId[8192];
int iSyncId[8192];
char szInetServer[256];
char szSqlServer[256];

/* FUNCTION: int main(int argc, char *argv[])
 *
 * PURPOSE: This function is the beginning execution point for the webcnct executable.
 *
 * ARGUMENTS:      int      argc      number of command line arguments passed to delivery
 *                char      *argv[]   array of command line argument pointers
 *
 * RETURNS:        None
 *
 * COMMENTS:       None
 *
 */

int main(int argc, char *argv[])
{
    HINTERNET hInternet;
    int iRc;
    int i;

    PrintHeader();
    if ( GetParameters(argc, argv) )
    {
        PrintParameters();
        return 0;
    }

    printf(szCmd1, "http://%/s/tpcc.dll?CMD=Begin&Server=%s&", szInetServer, szSqlServer);
    printf(szCmd2, "http://%/s/tpcc.dll?FORMID=1&w_id=1&d_id=6&CMD=Submit", szInetServer);
    printf(szCmd3, "http://%/s/tpcc.dll?STATUSID=0&FORMID=2&TERMID=%d&SYNCID=%d&CMD=..Exit..", szInetServer);

    printf("About to attempt %d connections.\n", iTotalConnections);
    printf("On Internet Server %s.\n", szInetServer);
    printf("On SQL Server %s.\n", szSqlServer);

    paktc();

    if ( !hInternet = InternetOpen("Web Test Connections", INTERNET_OPEN_TYPE_DIRECT, NULL, NULL, 0) )
    {
        printf("Error cannot open internet connection, GetLastError() = %d\n",
            GetLastError());
        return 0;
    }

    for(i=0; i<iTotalConnections; i++)
    {
        if ( (iRc = Login(hInternet, &iTermId[i], &iSyncId[i])) )
        {
            printf("Error = %d\n", iRc);
            goto main_exit;
        }
        printf("Login() TermID = %d SyncID = %d\n", iTermId[i], iSyncId[i]);
    }

    printf("All %d Connections made.\n", iTotalConnections);
    paktc();

    for(i=0; i<iTotalConnections; i++)
    {
        printf("Logout() TermID = %d SyncID = %d\n", iTermId[i], iSyncId[i]);
        if ( (iRc = Logout(hInternet, iTermId[i], iSyncId[i])) )
        {
            printf("Error = %d\n", iRc);
            goto main_exit;
        }
    }

main_exit:

    InternetCloseHandle(hInternet);

    return 0;
}

/* FUNCTION: void paktc(void)
 *
 * PURPOSE: This function displays a message on the console and waits for the user to press akey.
 *
 * ARGUMENTS:      none
 *
 * RETURNS:        None
 */
```

## Appendix A - Application Source Code

---

```
*
* COMMENTS:      None
*
*/

void paktc(void)
{
    printf("Press any Key to Continue.");
    getch();
}

/* FUNCTION: int Login(HINTERNET hInternet, int *pTermId, int *pSyncId)
*
* PURPOSE: This function logs the specified number of clients into the IIS TPCC.DLL client.
*
* ARGUMENTS:      HINTERNET hInternet      internet handle return from InternetOpen()
*                  int                iTermId      terminal id array
*                  int                iSyncId     sync id array
*
* RETURNS:        0 if successfull or error code if an error occurs.
*
* COMMENTS:       The TermID and SyncID returned are provided to the logout function
*                  on exit or logout.
*
*/

int Login(HINTERNET hInternet, int *pTermId, int *pSyncId)
{
    int                iRc;
    int                iPageSize;
    char               *ptr;
    char               *pBuffer;
    static char *szKeyTermId = "NAME=\\\"TERMINID\\\" VALUE=\\\"";
    static char *szKeySyncId = "NAME=\\\"SYNCID\\\" VALUE=\\\"";

    if ( (iRc = ReadUrlPage(hInternet, szCmd1, &pBuffer, NULL)) )
    {
        printf("ERROR: %s\n", pBuffer);
        return iRc;
    }
    free(pBuffer);

    if ( (iRc = ReadUrlPage(hInternet, szCmd2, &pBuffer, &iPageSize)) )
    {
        printf("ERROR: %s\n", pBuffer);
        return iRc;
    }
    ptr = strstr(pBuffer, szKeyTermId);
    if ( ptr )
    {
        ptr += strlen(szKeyTermId);
        *pTermId = atoi(ptr);

        ptr = strstr(pBuffer, szKeySyncId);
        if ( ptr )
        {
            ptr += strlen(szKeySyncId);
            *pSyncId = atoi(ptr);
            free(pBuffer);
            return 0;
        }
    }
    free(pBuffer);
    return -1;
}

/* FUNCTION: int Logout(HINTERNET hInternet, int iTermId, int iSyncId)
*
* PURPOSE: This function logs the number of clients logged into the IIS TPCC.DLL client out.
*
* ARGUMENTS:      HINTERNET hInternet      internet handle return from InternetOpen()
*                  int                iTermId      terminal id array, provided from Login API.
*                  int                iSyncId     sync id array, provided from Login API.
*
* RETURNS:        0 if successfull or error code if an error occurs.
*
* COMMENTS:       None
*
*/

int Logout(HINTERNET hInternet, int iTermId, int iSyncId)
{
    char               *pBuffer;
    char               szTmp[256];
    int                iRc;

    sprintf(szTmp, szCmd3, iTermId, iSyncId);

    iRc = ReadUrlPage(hInternet, szTmp, &pBuffer, NULL);
    free(pBuffer);
}
```



## Appendix A - Application Source Code

---

```
        return iRe;
    }

/* FUNCTION: int ReadUrlPage(HINTERNET hInternet, char *szUrl, char **ppBuffer, int *piTotalBytes)
 *
 * PURPOSE: This function logs the number of clients logged into the IIS TPCC.DLL client out.
 *
 * ARGUMENTS:      HINTERNET hInternet    internet handle return from InternetOpen()
 *                 char          *szUrl      URL string passed to the internet service
 *                 char          **ppBuffer  pointer to dynamically allocated buffer containing
 *                                           the result HTML page read
 *
 *                 int            *piTotalBytes size of the returned HTML page in bytes.
 *
 * RETURNS:        0 if successfull or error code if an error occurs.
 *
 * COMMENTS:       The caller is responsible for freeing the returned buffer.
 *
 */

int ReadUrlPage(HINTERNET hInternet, char *szUrl, char **ppBuffer, int *piTotalBytes)
{
    HINTERNET hUrl;
    BOOL      bOk;
    int       bytesRead;
    int       iBuffSize;
    int       iCurrentByte;
    char      szTmp[512];

    *ppBuffer = (char *)malloc(512);
    iBuffSize = 512;
    iCurrentByte = 0;
    **ppBuffer = 0;
    if ( piTotalBytes )
        *piTotalBytes = 0;

    hUrl = InternetOpenUrl(hInternet, szUrl, NULL, 0, INTERNET_FLAG_RELOAD, 0);
    if ( !hUrl )
    {
        return GetLastError();
    }
    do
    {
        bOk = InternetReadFile(hUrl, szTmp, sizeof(szTmp), &bytesRead);
        if ( !bOk )
            break;
        if ( (iBuffSize - iCurrentByte) <= bytesRead )
        {
            iBuffSize += ((bytesRead/512)+1)*512;
            *ppBuffer = realloc(*ppBuffer, iBuffSize );
            if ( !(*ppBuffer) )
            {
                InternetCloseHandle(hUrl);
                return -2;
            }
        }
        memcpy((*ppBuffer)+iCurrentByte, szTmp, bytesRead);
        iCurrentByte += bytesRead;
    } while( bOk && (bytesRead != 0) );

    InternetCloseHandle(hUrl);

    if ( !bOk )
        return GetLastError();

    if ( piTotalBytes )
        *piTotalBytes = iCurrentByte;

    return 0;
}

/* FUNCTION: BOOL GetParameters(int argc, char *argv[])
 *
 * PURPOSE: This function parses the command line passed in to the delivery executable, initializing
 *          and filling in global variable parameters.
 *
 * ARGUMENTS:      int          argc      number of command line arguments passed to delivery
 *                 char          *argv[]  array of command line argument pointers
 *
 * RETURNS:        BOOL        FALSE     parameter read successful
 *                 TRUE         user has requested parameter information screen be displayed.
 *
 * COMMENTS:       None
 *
 */

static BOOL GetParameters(int argc, char *argv[])
{

```

## Appendix A - Application Source Code

---

```
int i;

szSqlServer[0] = 0;
szInetServer[0] = 0;
iTotalConnections = 100;

for(i=0; i<argc; i++)
{
    if ( argv[i][0] == '-' || argv[i][0] == '/' )
    {
        switch(argv[i][1])
        {
            case 'S':
            case 's':
                strcpy(szSqlServer, argv[i+2]);
                break;

            case 'I':
            case 'i':
                strcpy(szInetServer, argv[i+2]);
                break;

            case 'C':
            case 'c':
                if ( !iTotalConnections = atoi(argv[i+2]))
                    iTotalConnections = 100;
                break;

            case '?':
                return TRUE;
        }
    }
}

if ( !szInetServer[0] )
    return TRUE;

return FALSE;
}

/* FUNCTION: void PrintParameters(void)
 *
 * PURPOSE: This function displays the supported command line flags.
 *
 * ARGUMENTS:      None
 *
 * RETURNS:        None
 *
 * COMMENTS:       None
 *
 */

static void PrintParameters(void)
{
    printf("WEBTEST:\n");
    printf("Parameter                                Default\n");
    printf("-----\n");
    printf("-S SQL Server for test run.                local \n");
    printf("-I Internet server for test run.(Required)  none \n");
    printf("-C Total connections for test run.          100 \n");
    printf("-? This help screen\n");
    printf("Note: Command line switches are NOT case sensitive.\n");

    return;
}

/* FUNCTION: void PrintHeader(void)
 *
 * PURPOSE: This function displays the delivery executable's banner information.
 *
 * ARGUMENTS:      None
 *
 * RETURNS:        None
 *
 * COMMENTS:       None
 *
 */

static void PrintHeader(void)
{
    printf("*****\n");
    printf("**                               *\n");
    printf("** Microsoft SQL Server 6.5         *\n");
    printf("**                               *\n");
    printf("** HTML TPC-C BENCHMARK KIT: Web Connect Utility *\n");
    printf("** Version %d.%2d.%3d               *\n", versionMS, versionMM, versionLS);
    printf("**                               *\n");
    printf("*****\n");

    return;
}
}
```

## Appendix B - Database Design

### Build

#### Createdb.sql

```
/* TPC-C Benchmark Kit */
/* */
/* CREATEDB.SQL */
/* This script is used to create the database */

use master
go

if exists ( select name from sysdatabases where name = "tpcc" )
    drop database tpcc
go

create database tpcc on

        bigdev1 = 1481,
        bigdev2 = 1481,
        bigdev3 = 1481,
        bigdev4 = 1481,
        bigdev5 = 1481,

        bigdev5 = 1481,
        bigdev1 = 1481,
        bigdev2 = 1481,
        bigdev3 = 1481,
        bigdev4 = 1481,

        bigdev4 = 1481,
        bigdev5 = 1481,
        bigdev1 = 1481,
        bigdev2 = 1481,
        bigdev3 = 1481,

        bigdev3 = 1481,
        bigdev4 = 1481,
        bigdev5 = 1481,
        bigdev1 = 1481,
        bigdev2 = 1481,

        bigdev2 = 1481,
        bigdev3 = 1481,
        bigdev4 = 1481,
        bigdev5 = 1481,
        bigdev1 = 1481,

        ol1 = 17178,
        misc1 = 13608

    log on tpcclog1 = 8670,
        tpcclog2 = 8670
go
```

#### Diskinit.sql

```
/* TPC-C Benchmark Kit */
/* */
/* DISKINIT.SQL */
/* This script is used create the database devices for a 500 */
/* warehouse database. */
/* NOTE! This version of DISKINIT.SQL assumes that you are using */
/* some form of NT partitioning. If you wish to use raw */
/* partitions, YOU MUST SPECIFY A DRIVE LETTER ONLY for the */
/* physname parm! Raw partitions will not accept a file name. */
/* Also note that use of a drive letter only for the physname */
/* parm will result in corruption of any normal NT partition! */

use master
go

disk init name = "tpcclog1",
        physname = "X:",
        vdevno = 2,
        size = 4442112
go
```

```
disk init name = "tpcclog2",
        physname = "V:",
        vdevno = 3,
        size = 4442112
go

disk init name = "tpcclog3",
        physname = "W:",
        vdevno = 4,
        size = 2048000
go

/*
 * Devices for Warehouse, District, Item, New_order, History, and Order tables
 */
disk init name = "misc1",
        physname = "E:",
        vdevno = 5,
        size = 6967296
go

/*
 * Devices for Order_line table
 */
disk init name = "ol1",
        physname = "F:",
        vdevno = 6,
        size = 8795136
go

/*
 * Devices for customer and stock tables
 */
disk init name = "bigdev1",
        physname = "G:",
        vdevno = 7,
        size = 4080640
go

disk init name = "bigdev2",
        physname = "H:",
        vdevno = 8,
        size = 4080640
go

disk init name = "bigdev3",
        physname = "I:",
        vdevno = 9,
        size = 4080640
go

disk init name = "bigdev4",
        physname = "J:",
        vdevno = 10,
        size = 4080640
go

disk init name = "bigdev5",
        physname = "K:",
        vdevno = 11,
        size = 4080640
go
```

#### Segment.sql

```
/* TPC-C Benchmark Kit */
/* */
/* SEGMENT.SQL */
/* This script is used to create the database segments */

use tpcc
go

/* Drop all segments if needed */

sp_dropsegment big_seg
go
sp_dropsegment ol_seg
go
sp_dropsegment misc_seg
go
```

## Appendix B - Database Design

```
/* Create segment for Customer and Stock tables */
```

```
exec sp_addsegment big_seg, bigdev1
exec sp_extendsegment big_seg, bigdev2
exec sp_extendsegment big_seg, bigdev3
exec sp_extendsegment big_seg, bigdev4
exec sp_extendsegment big_seg, bigdev5
go
```

```
/* Create segment for Order-line table */
```

```
exec sp_addsegment ol_seg, ol1
go
```

```
/* create segment for warehouse, district, item, new-order, history, and Order
tables */
```

```
exec sp_addsegment misc_seg, misc1
go
```

### Tables.sql

```
/* TPC-C Benchmark Kit          */
/*                               */
/* TABLES.SQL                  */
/*                               */
/* Creates TPC-C tables (seg)   */
/*                               */
```

```
use tpcc
go
```

```
checkpoint
go
```

```
if exists ( select name from sysobjects where name = 'warehouse' )
drop table warehouse
```

```
go
```

```
create table warehouse
```

```
(
    w_id                tinyint,
    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 misc_seg
```

```
go
```

```
if exists ( select name from sysobjects where name = 'district' )
drop table district
```

```
go
```

```
create table district
```

```
(
    d_id                tinyint,
    d_w_id              tinyint,
    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 misc_seg
```

```
go
```

```
if exists ( select name from sysobjects where name = 'customer' )
drop table customer
```

```
go
```

```
create table customer
```

```
(
    c_id                int,
    c_d_id              tinyint,
    c_w_id              smallint,
    c_first            char(16),
    c_middle            char(2),
    c_last             char(16),
    c_street_1         char(20),
    c_street_2         char(20),
    c_city             char(20),
    c_state            char(2),
    c_zip              char(9),
    c_phone            char(16),
    c_since            datetime,
    c_credit            char(2),
    c_credit_lim       numeric(12,2),
    c_discount         numeric(4,4),
    c_balance          numeric(12,2),
    c_ytd_payment     numeric(12,2),
    c_payment_cnt     smallint,
    c_delivery_cnt    smallint,
    c_data_1           char(250),
    c_data_2           char(250)
) on big_seg
```

```
go
```

```
if exists ( select name from sysobjects where name = 'history' )
drop table history
```

```
go
```

```
create table history
```

```
(
    h_c_id              int,
    h_c_d_id            tinyint,
    h_c_w_id            smallint,
    h_d_id              tinyint,
    h_w_id              tinyint,
    h_date              smallint,
    h_date              datetime,
    h_amount            numeric(6,2),
    h_data              char(24)
) on misc_seg
```

```
go
```

```
if exists ( select name from sysobjects where name = 'new_order' )
drop table new_order
```

```
go
```

```
create table new_order
```

```
(
    no_o_id             int,
    no_d_id             tinyint,
    no_w_id             tinyint,
    no_line_id          smallint
) on misc_seg
```

```
go
```

```
if exists ( select name from sysobjects where name = 'orders' )
drop table orders
```

```
go
```

```
create table orders
```

```
(
    o_id               int,
    o_d_id             tinyint,
    o_w_id             tinyint,
    o_line_id          smallint,
    o_c_id             int,
    o_entry_d          datetime,
    o_carrier_id       tinyint,
    o_ol_cnt           tinyint,
    o_all_local        tinyint
) on misc_seg
```

```
go
```

## Appendix B - Database Design

```
go

if exists ( select name from sysobjects where name = 'order_line' )
    drop table order_line
go

create table order_line
(
    ol_o_id          int,
    ol_d_id          tinyint,
    ol_w_id          smallint,
    ol_number        tinyint,
    ol_i_id          int,
    ol_supply_w_id   smallint,
    ol_delivery_d     datetime,
    ol_quantity       smallint,
    ol_amount         numeric(6,2),
    ol_dist_info     char(24)
) on ol_seg
go

if exists ( select name from sysobjects where name = 'item' )
    drop table item
go

create table item
(
    i_id             int,
    i_im_id          int,
    i_name           char(24),
    i_price          numeric(5,2),
    i_data           char(50)
) on misc_seg
go

if exists ( select name from sysobjects where name = 'stock' )
    drop table stock
go

create table stock
(
    s_i_id          int,
    s_w_id          smallint,
    s_quantity       smallint,
    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 big_seg
go
```

### Idxcuscl.sql

```
/* TPC-C Benchmark Kit          */
/*                               */
/* IDXCUSCL.SQL                  */
/*                               */
/* Creates clustered index on customer (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'customer_c1' )
    drop index customer.customer_c1
go

select getdate()
go
create unique clustered index customer_c1 on customer(c_w_id, c_d_id, c_id)
```

```
with sorted_data on big_seg
go
select getdate()
go

/* TPC-C Benchmark Kit          */
/*                               */
/* IDXCUSNC.SQL                  */
/*                               */
/* Creates non-clustered index on customer (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'customer_nc1' )
    drop index customer.customer_nc1
go

select getdate()
go
create unique nonclustered index customer_nc1 on customer(c_w_id, c_d_id,
c_last, c_first, c_id)
on big_seg
go
select getdate()
go
```

### Idxdiscl.sql

```
/* TPC-C Benchmark Kit          */
/*                               */
/* IDXDISCL.SQL                  */
/*                               */
/* Creates clustered index on district (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'district_c1' )
    drop index district.district_c1
go

select getdate()
go
create unique clustered index district_c1 on district(d_w_id, d_id)
with fillfactor=1 on misc_seg
go
select getdate()
go
```

### Idxitmcl.sql

```
/* TPC-C Benchmark Kit          */
/*                               */
/* IDXITMCL.SQL                  */
/*                               */
/* Creates clustered index on item (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'item_c1' )
    drop index item.item_c1
go

select getdate()
go
create unique clustered index item_c1 on item(i_id)
with sorted_data on misc_seg
go
select getdate()
go
```

### Idxnodcl.sql

```
/* TPC-C Benchmark Kit          */
/*                               */
/* IDXNODCL.SQL                  */
/*                               */

select getdate()
go
```

## Appendix B - Database Design

---

```
/* Creates clustered index on new-order (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'new_order_c1' )
    drop index new_order.new_order_c1
go

select getdate()
go
create unique clustered index new_order_c1 on new_order(no_w_id, no_d_id,
no_o_id)
    with sorted_data on misc_seg
go
select getdate()
go
```

### Idxodlcl.sql

```
/* TPC-C Benchmark Kit */
/* IDXDLC.LSQL */
/* Creates clustered index on order-line (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'order_line_c1' )
    drop index order_line.order_line_c1
go

select getdate()
go
create unique clustered index order_line_c1 on order_line(ol_w_id, ol_d_id,
ol_o_id, ol_number)
    with sorted_data on ol_seg
go
select getdate()
go
```

### Idxordcl.sql

```
/* TPC-C Benchmark Kit */
/* IDXORDCL.SQL */
/* Creates clustered index on orders (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'orders_c1' )
    drop index orders.orders_c1
go

select getdate()
go
create unique clustered index orders_c1 on orders(o_w_id, o_d_id, o_id)
    with sorted_data on misc_seg
go
select getdate()
go
```

### Idxstkcl.sql

```
/* TPC-C Benchmark Kit */
/* IDXSTKCL.SQL */
/* Creates clustered index on stock (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'stock_c1' )
    drop index stock.stock_c1
go
```

```
select getdate()
go
create unique clustered index stock_c1 on stock(s_i_id, s_w_id)
    with sorted_data on big_seg
go
select getdate()
go
```

### Idxwarcl.sql

```
/* TPC-C Benchmark Kit */
/* IDXWARCL.SQL */
/* Creates clustered index on warehouse (seg) */

use tpcc
go

if exists ( select name from sysindexes where name = 'warehouse_c1' )
    drop index warehouse.warehouse_c1
go

select getdate()
go
create unique clustered index warehouse_c1 on warehouse(w_id)
    with fillfactor=1 on misc_seg
go
select getdate()
go
```

### Dbopt1.sql

```
/* TPC-C Benchmark Kit */
/* DBOPT1.SQL */
/* Set database options for database load */

use master
go

sp_dboption tpcc,'select into/bulkcopy',true
go

sp_dboption tpcc,'trunc. log on chkpt.',true
go

use tpcc
go

checkpoint
go

use tpcc_admin
go

sp_dboption tpcc,'trunc. log on chkpt.',true
go
```

### Dbopt2.sql

```
/* TPC-C Benchmark Kit */
/* DBOPT2.SQL */
/* Reset database options after database load */

use master
go

sp_dboption tpcc,'select ',false
go

sp_dboption tpcc,'trunc. ',false
go

use tpcc
go

checkpoint
go
```

# Appendix B - Database Design

## Pintable.sql

```
/* TPC-C Benchmark Kit */
/* PINTABLE.SQL */
/* This script file is used to 'pin' certain tables in the data cache */

use tpcc
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
```

## Tpccbcpl.sql

```
/* TPC-C Benchmark Kit */
/* TPCCBCP.SQL */
/* This script file sets the table lock option for bulk load */

use tpcc
go

exec sp_tableoption "warehouse","table lock on bulk load",true
exec sp_tableoption "district","table lock on bulk load",true
exec sp_tableoption "stock","table lock on bulk load",true
exec sp_tableoption "item","table lock on bulk load",true
exec sp_tableoption "customer","table lock on bulk load",true
exec sp_tableoption "history","table lock on bulk load",true
exec sp_tableoption "orders","table lock on bulk load",true
exec sp_tableoption "order_line","table lock on bulk load",true
exec sp_tableoption "new_order","table lock on bulk load",true
go
```

## Tpccirl.sql

```
/* TPC-C Benchmark Kit */
/* TPCCIRL.SQL */
/* This script file sets the insert row lock option on selected tables */

use tpcc
go

exec sp_tableoption "history","insert row lock",true
exec sp_tableoption "new_order","insert row lock",true
exec sp_tableoption "orders","insert row lock",true
exec sp_tableoption "order_line","insert row lock",true
go
```

## Stored Procedures

### Neworder.sql

```
/* File: NEWORDER.SQL */
/* Microsoft TPC-C Kit Ver. 3.00.000 */
/* Audited 08/23/96, By Francois Raab */
/* Copyright Microsoft, 1996 */
/* Purpose: New-Order transaction for Microsoft TPC-C Benchmark Kit */
/* Author: Damien Lindauer */
/* damienl@Microsoft.com */

use tpcc
go

/* new-order transaction stored procedure */

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 order date */
select @o_entry_d = getdate()
/* get district tax and next available order id and update */
update district
set @d_tax = d_tax,
@o_id = d_next_o_id,
d_next_o_id = d_next_o_id + 1
where d_w_id = @w_id and
d_id = @d_id
/* process orderlines */
select @li_no = 0
/* set commit flag */
select @commit_flag = 1
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
```

## Appendix B - Database Design

```

when 15 then @i_id15
end

select @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

select @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 holdlock)
where i_id = @li_id

/* if there actually is an item with this id, go to work */

if (@@rowcount > 0)
begin
update stock set s_ytd = s_ytd + @li_qty,
               @s_quantity = 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

/* insert order_line data (using data
from item and stock) */

insert into order_line values(@o_id, /* from district update */
                             @d_id, /* input param
                             @w_id, /* input param */

                             @li_no, /* orderline number */
                             @li_id, /* lineitem id */
                             @li_s_w_id, /* lineitem warehouse */
                             "jan 1, 1900", /* constant */
                             @li_qty, /* lineitem qty */
                             @i_price * @li_qty, /* ol_amount */
                             @s_dist) /* from stock */

/* 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 found - triggers rollback condition */

select "", "", 0.0
select @commit_flag = 0

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 holdlock
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 holdlock
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

/* File: PAYMENT.SQL */

```

### Payment.sql

/\* File: PAYMENT.SQL

\*/



## Appendix B - Database Design

```
/* Microsoft TPC-C Kit Ver. 3.00.000 */
/* Audited 08/23/96, By Francois Raab */
/* Copyright Microsoft, 1996 */
/* Purpose: Payment transaction for Microsoft TPC-C Benchmark Kit */
/* Author: Damien Lindauer */
/* damienl@Microsoft.com */

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),
        @data1 char(250),
        @data2 char(250),
        @c_data_1 char(250),
        @c_data_2 char(250),
        @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 holdlock
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 holdlock
where c_last = @c_last and
c_w_id = @c_w_id and
c_d_id = @c_d_id
order by c_w_id, c_d_id, c_last, c_first

set rowcount 0

end

/* get customer info and update balances */

update customer set
@c_balance = c_balance -
@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,
@data1 = @c_data_1,
@data2 = @c_data_2,
@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_2 = substring(@data1,209,42) +
substring(@data2, 1,
208)

select @c_data_1 = convert(char(5),@c_id) +
convert(char(4),@c_d_id) +
convert(char(5),@c_w_id) +
convert(char(4),@d_id) +
convert(char(5),@w_id) +
convert(char(19),@h_amount) +
substring(@data1, 1, 208)

/* update customer info */

update customer set
c_data_1 = @c_data_1,
c_data_2 = @c_data_2
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_1,1,200)

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
```

## Appendix B - Database Design

```
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 Kit Ver. 3.00.000 */
/* Audited 08/23/96, By Francois Raab */
/* Copyright Microsoft, 1996 */
/* Purpose: Delivery transaction for Microsoft TPC-C Benchmark Kit */
/* Author: Damien Lindauer */
/* damienl@Microsoft.com */

use tpcc
go

/* delivery transaction */

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 @o_id = min(no_o_id)
    from new_order holdlock
    where no_w_id = @w_id and
        no_d_id = @d_id

    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

        /* accumulate 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

select @oid1,
    @oid2,
    @oid3,
    @oid4,
    @oid5,
    @oid6,
    @oid7,
    @oid8,
    @oid9,
```

## Appendix B - Database Design

```
@oid10

go

Ordstat.sql

/* File:   ORDSTAT.SQL
/* Microsoft TPC-C Kit Ver. 3.00.000
/* Audited 08/23/96, By Francois Raab
/*
/* Copyright Microsoft, 1996
/*
/* Purpose: Order-Status transaction for Microsoft TPC-C Benchmark Kit
/* Author:   Damien Lindauer
/* damienl@Microsoft.com

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,
        @val               smallint,
        @cnt               smallint

begin tran o

if (@c_id = 0)
begin
/* get customer id and info using last name */

select @cnt = count(*)
from customer holdlock
where c_last = @c_last and
c_w_id = @w_id and
c_d_id = @d_id

select @val = (@cnt + 1) / 2
set rowcount @val

select @c_id = c_id,
       @c_balance = c_balance,
       @c_first = c_first,
       @c_last = c_last,
       @c_middle = c_middle
from customer holdlock
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 holdlock
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 holdlock
where o_c_id = @c_id and
o_d_id = @d_id and
o_w_id = @w_id

/* 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 holdlock
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
```

## Stocklev.sql

```
/* File:   STOCKLEV.SQL
/* Microsoft TPC-C Kit Ver. 3.00.000
/* Audited 08/23/96, By Francois Raab
/* Copyright Microsoft, 1996
/* Purpose: Stock-Level transaction for Microsoft TPC-C Benchmark Kit
/* Author:   Damien Lindauer
/* damienl@Microsoft.com

use tpcc
go

/* stock-level transaction stored procedure */

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,
                             @threshold    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 < @threshold
```

go

### Appendix C - Tuning Parameters

#### *Microsoft Windows NT Server Tunable Parameters*

##### Registry Changes

One Windows NT registry parameter on the EnVista Server was modified for this benchmark:

Key Name: SYSTEM\ControlSet001\Services\aic78xx\Parameters\Device  
Class Name: <NO CLASS>  
Last Write Time: 3/17/97 - 6:36 PM  
Value 0  
Name: NumberOfRequests  
Type: REG\_DWORD  
Data: 0x30

This modification set the number of outstanding requests on the Adaptec AIC-78xx controllers to 48 (decimal).

##### Microsoft SQL Server version 6.5 Startup Parameters

Microsoft SQL Server was started with the following command line options

```
sqlservr -c -x -t1081 -t3502
```

where

-c	Start SQL Server independently of the Microsoft Windows NT Service Control Manager.
-x	Disable the keeping of CPU time and cache-hit ration statistics.
-t1081	Allow the index pages a second trip through cache before those pages are released.
-t3502	Prints a message to the log at the beginning and end of each checkpoint.

##### DBCC GAMINIT

The Global Allocation Map (GAM) was pre-populated using DBCC GAMINIT prior to starting the benchmark.

##### Microsoft Windows NT Server Configuration Parameters

The following services were disabled on the Server prior to benchmark execution:

- Computer Browser
- License Manager
- Messenger
- NT LM Security Provider
- Plug and Play
- TCP/IP NetBIOS Helper
- Spooler

## Appendix C - Tuning Parameters

---

### Microsoft SQL Server 6.5 Configuration Parameters

name	minimum	maximum	config_value	run_value
affinity mask	0	2147483647	0	0
allow updates	0	1	0	0
backup buffer size	1	32	1	1
backup threads	0	32	5	5
cursor threshold	-1	2147483647	-1	-1
database size	2	10000	2	2
default language	0	9999	0	0
default sortorder id	0	255	50	50
fill factor	0	100	0	0
free buffers	20	524288	5000	5000
hash buckets	4999	265003	265003	265003
language in cache	3	100	3	3
LE threshold maximum	2	500000	200	200
LE threshold minimum	2	500000	20	20
LE threshold percent	1	100	0	0
locks	5000	2147483647	5000	5000
LogLRU buffers	0	2147483647	1500	1500
logwrite sleep (ms)	-1	500	0	0
max async IO	1	1024	32	32
max lazywrite IO	1	1024	24	24
max text repl size	0	2147483647	65536	65536
max worker threads	10	1024	100	100
media retention	0	365	0	0
memory	2800	1048576	870000	870000
nested triggers	0	1	1	1
network packet size	512	32767	3072	3072
open databases	5	32767	20	20
open objects	100	2147483647	500	500
priority boost	0	1	0	0
procedure cache	1	99	2	2
Protection cache size	1	8192	15	15
RA cache hit limit	1	255	4	4
RA cache miss limit	1	255	3	3
RA delay	0	500	15	15
RA pre-fetches	1	1000	3	3
RA slots per thread	1	255	5	5
RA worker threads	0	255	0	0
recovery flags	0	1	0	0
recovery interval	1	32767	32767	32767
remote access	0	1	0	0
remote conn timeout	-1	32767	10	10
remote login timeout	0	2147483647	5	5
remote proc trans	0	1	0	0
remote query timeout	0	2147483647	0	0
remote sites	0	256	0	0
resource timeout	5	2147483647	10	10
set working set size	0	1	0	0
show advanced options	0	1	1	1
SMP concurrency	-1	64	-1	-1
sort pages	64	511	64	64
spin counter	1	2147483647	10000	10000
tempdb in ram (MB)	0	2044	5	5
time slice	50	1000	100	100
user connections	5	32767	6350	6350
user options	0	4095	0	0

## Appendix C - Tuning Parameters

---

### Microsoft SQL Server Stack Size

The default stack size of Microsoft SQL Server was changed using the EDITBIN utility. The EDITBIN utility ships with Microsoft Visual C++ V4.2. The command used was `editbin /stack:32768 sqlserver.exe`.

### Disk Array Configuration Parameters

LUN	1	2	3	4	5	6
#Disks	15	15	15	15	15	15
RAID Level	0	0	0	0	0	0
Segment Size	4K	4K	4K	4K	4K	4K
Write Cache	enabled	enabled	enabled	enabled	enabled	enabled
Cache Mirror	disabled	disabled	disabled	disabled	disabled	disabled
Force Unit Access	on	on	on	on	on	on
Partitions	2	1	1	1	1	2

Force Unit Access is set in the LVS 4500 controllers to force Microsoft SQL Server 6.5 to utilize the caches in “write-to” mode. Cache mirroring is a reliability/failover feature of the LVS 4500 that is not utilized in the TPC benchmark.

### Client Configuration Parameters

The Client system configurations were modified as described below.

### Microsoft Windows NT Server Configuration Parameters

No NT parameters were changed on the client machines.

The following services were disabled on the client machines:

- Computer Browser
- FTP Publishing Service
- Gopher Publishing Service
- License Logging Service
- Messenger
- Net Logon
- NT LM Security Support Provider
- Plug and Play
- RPC Locator
- Schedule
- Spooler
- TCP/IP NetBIOS Helper
- UPS

## Appendix C - Tuning Parameters

---

### RTE Parameters

Profile: 6200users  
File Path: C:\rte\6200users.pro  
Version: 1.0.0

Number of Engines: 6

Name: RTE1  
Description: ACLIENT1  
Directory: C:\rte\arte1.log  
Machine: ARTE1  
Parameter Set: plus-10  
Index: 0  
Seed: 57543  
Configured Users: 1040  
Pipe Name: DRIVER111826846  
Connect Rate: 100  
Start Rate: 0  
CLIENT\_NURAND: 233

Name: RTE2  
Description: ACLIENT2  
Directory: C:\rte\arte2.log  
Machine: ARTE2  
Parameter Set: plus-10  
Index: 100000  
Seed: 57543  
Configured Users: 1040  
Pipe Name: DRIVER211882295  
Connect Rate: 100  
Start Rate: 0  
CLIENT\_NURAND: 233

Name: RTE3  
Description: ACLIENT3  
Directory: C:\rte\arte3.log  
Machine: ARTE3  
Parameter Set: plus-10  
Index: 200000  
Seed: 57543  
Configured Users: 1040  
Pipe Name: DRIVER312000425  
Connect Rate: 100  
Start Rate: 0  
CLIENT\_NURAND: 233

Name: RTE4  
Description: ACLIENT4  
Directory: C:\rte\arte4.log  
Machine: ARTE4  
Parameter Set: plus-10  
Index: 300000



## Appendix C - Tuning Parameters

---

Seed: 57543  
Configured Users: 1040  
Pipe Name: DRIVER412025752  
Connect Rate: 100  
Start Rate: 0  
CLIENT\_NURAND: 233

Name: RTE5  
Description: ACLIENT5  
Directory: C:\rte\arte5.log  
Machine: ARTE5  
Parameter Set: plus-10  
Index: 400000  
Seed: 57543  
Configured Users: 1040  
Pipe Name: DRIVER512105106  
Connect Rate: 100  
Start Rate: 0  
CLIENT\_NURAND: 233

Name: RTE6  
Description: ACLIENT6  
Directory: C:\rte\arte6.log  
Machine: ARTE6  
Parameter Set: plus-10  
Index: 500000  
Seed: 57543  
Configured Users: 1000  
Pipe Name: DRIVER613784941  
Connect Rate: 100  
Start Rate: 0  
CLIENT\_NURAND: 233

Number of Users: 6

Driver Engine: RTE1  
IIS Server: ACLIENT1  
SQL Server: ENVISTA  
User: sa  
Protocol: Html  
w\_id Range: 1 - 104  
w\_id Max Warehouse: 620  
Scale: Normal  
User Count: 1040  
District id: 1  
Scale Down: No

Driver Engine: RTE2  
IIS Server: ACLIENT2  
SQL Server: ENVISTA  
User: sa  
Protocol: Html  
w\_id Range: 105 - 208  
w\_id Max Warehouse: 620  
Scale: Normal

## Appendix C - Tuning Parameters

---

User Count: 1040  
District id: 1  
Scale Down: No

Driver Engine: RTE3  
IIS Server: ACLIENT3  
SQL Server: ENVISTA  
User: sa  
Protocol: Html  
w\_id Range: 209 - 312  
w\_id Max Warehouse: 620  
Scale: Normal  
User Count: 1040  
District id: 1  
Scale Down: No

Driver Engine: RTE4  
IIS Server: ACLIENT4  
SQL Server: ENVISTA  
User: sa  
Protocol: Html  
w\_id Range: 313 - 416  
w\_id Max Warehouse: 620  
Scale: Normal  
User Count: 1040  
District id: 1  
Scale Down: No

Driver Engine: RTE5  
IIS Server: ACLIENT5  
SQL Server: ENVISTA  
User: sa  
Protocol: Html  
w\_id Range: 417 - 520  
w\_id Max Warehouse: 620  
Scale: Normal  
User Count: 1040  
District id: 1  
Scale Down: No

Driver Engine: RTE6  
IIS Server: ACLIENT6  
SQL Server: ENVISTA  
User: sa  
Protocol: Html  
w\_id Range: 521 - 620  
w\_id Max Warehouse: 620  
Scale: Normal  
User Count: 1000  
District id: 1  
Scale Down: No

## Appendix C - Tuning Parameters

---

### RTE Pacing Sets

The following BenchCraft RTE pacing sets were used in the benchmark. The measured tpmC was established using pacing set PLUS-10.

#### PLUS 1

Txn	Think	Key	RT	RT	Menu		
Weight	Time	Time	Delay	Fence	Delay		
New Order	44.72		12.98	18.02	0.10	5.00	0.10
Payment	43.08		12.97	3.03	0.10	5.00	0.10
Delivery	4.04		5.97	2.03	0.10	5.00	0.10
Stock Level	4.08		5.97	2.03	0.10	20.00	0.10
Order Status	4.08		10.97	2.03	0.10	5.00	0.10

#### PLUS 2

Txn	Think	Key	RT	RT	Menu		
Weight	Time	Time	Delay	Fence	Delay		
New Order	44.72		13.98	18.02	0.10	5.00	0.10
Payment	43.08		13.97	3.03	0.10	5.00	0.10
Delivery	4.04		6.97	3.03	0.10	5.00	0.10
Stock Level	4.08		6.97	3.03	0.10	20.00	0.10
Order Status	4.08		11.97	3.03	0.10	5.00	0.10

#### ~Default

##### Default Parameter Set

Txn	Think	Key	RT	RT	Menu		
Weight	Time	Time	Delay	Fence	Delay		
New Order	44.72		12.00	18.02	0.10	5.00	0.10
Payment	43.08		12.00	3.03	0.10	5.00	0.10
Delivery	4.04		5.00	2.03	0.10	5.00	0.10
Stock Level	4.08		5.00	2.03	0.10	20.00	0.10
Order Status	4.08		10.00	2.03	0.10	5.00	0.10

#### 2 X Defaults

Txn	Think	Key	RT	RT	Menu		
Weight	Time	Time	Delay	Fence	Delay		
New Order	44.80		24.00	18.00	0.10	5.00	0.10
Payment	43.05		24.00	3.00	0.10	5.00	0.10
Delivery	4.05		10.00	2.00	0.10	5.00	0.10
Stock Level	4.05		10.00	2.00	0.10	20.00	0.10
Order Status	4.05		20.00	2.00	0.10	5.00	0.10

#### PLUS-10

Txn	Think	Key	RT	RT	Menu		
Weight	Time	Time	Delay	Fence	Delay		
New Order	44.72		12.08	18.02	0.10	5.00	0.10
Payment	43.08		12.07	3.03	0.10	5.00	0.10
Delivery	4.04		5.07	2.03	0.10	5.00	0.10
Stock Level	4.08		5.07	2.03	0.10	20.00	0.10
Order Status	4.08		10.07	2.03	0.10	5.00	0.10

## Appendix C - Tuning Parameters

---

PLUS -05

	Txn	Think	Key	RT	RT	Menu	
	Weight	Time	Time	Delay	Fence	Delay	
New Order	44.68		12.03	18.02	0.10	5.00	0.10
Payment	43.08		12.02	3.03	0.10	5.00	0.10
Delivery	4.08		5.02	2.03	0.10	5.00	0.10
Stock Level	4.08		5.02	2.03	0.10	20.00	0.10
Order Status	4.08		10.02	2.03	0.10	5.00	0.10

# Appendix D - Disk Storage

## Appendix D - Disk Storage

### 180 Day Space Calculations

<b>Warehouses</b>	620				
<b>TpmC</b>	7573				
<b>Table</b>	<b>Rows</b>	<b>Data 1K pages</b>	<b>Index 1K pages</b>	<b>Extra 5%</b>	<b>Total with 5%</b>
Warehouse	620	1,300	10	66	1,376
District	6,200	13,000	56	653	13,709
Item	100,000	9,100	46	457	9,603
Customer	18,600,000	13,002,600	1,009,188	700,589	14,712,377
New_Order	5,580,000	65,000	398	3,270	68,668
Stock	62,000,000	21,671,000	119,736	1,089,537	22,880,273
History (Dynamic)	18,600,000	975,002	0		975,002
Orders (Dynamic)	18,600,000	507,000	3,060		510,060
Order_line (Dynamic)	195,004,080	10,839,880	70,856		10,910,736
<b>Total MB</b>	<b>318,490,900</b>	<b>47,083,882</b>	<b>1,203,350</b>	<b>1,794,572</b>	<b>50,081,804</b>
<b>Loaded</b>	48,287,232				
<b>With 5%</b>	50,081,804				
<b>8 Hours</b>					
<b>Dabase Layout</b>	<b># of Segs</b>	<b>Size in MB</b>	<b>Total Allocated</b>	<b>Tables</b>	
Master	1	17	17	17	
Model	1	1	1	1	
MsdB	1	8	8	8	Allocated
CS_Dev (big)	4	8,427	33,708	c,s	37,921
CS_Dev (small)	1	4,213	4,213		36,712
Misc_dev (big)	4	1,404	5,616	w,d,l,no,h,o	1,541
Misc_dev (small)	1	702	702		
OL_dev	2	8,450	16,900	ol	10,655
Total Allocated					61,165
					48,908
	<b>MB</b>				
Dynamic Space	12,395,798	Sum of Data + bitmap for Order, Order_Line and History			
Static Space	37,686,006	Sum of all data, index, bitmap (including the rootdbs) +5% - above dynamic space			
Free Space	12,551,156	Total space allocated to DBMS - dynamic and static			
Daily growth	2,422,538.79	(Dynamic space / (W * 62.5)) * tpmC			
Daily spread	8,917,348.11	Free space - 1.5 * Daily growth (zero if negative)			
	0	This can be reconfigured to eliminate daily spread, zero assumed			
180 day space (MB)	473,742,989	Static space + 180 * (daily growth + daily spread)			
180 day space (GB)	<b>462,640</b>				
8 hr log space (GB)	<b>21.16</b>	(excludes RAID-1)			
					8.36706543
	<b>Space Needed</b>	<b>Disk size (GB)</b>	<b>Disks Priced</b>	<b>GB</b>	
180 day space	462,640	3.992	60	239.52	
		8.367	30	251.01	
				<u>490.53</u>	
Logical Logs (w/mirrors)	42.32	8.367	6	50.202	
OS, file sys, Swap	0.5				
<b>Total</b>	<b>462682.4573</b>				

# Appendix E - Price Quotes

## Appendix E - Price Quotes



**E. C. W. ENTERPRISE, INC.**  
*dba / HiQ Computer Systems*

740 North Mary Avenue  
 Sunnyvale, CA 94086  
 TEL: (408) 245-5836  
 (800) 827-5836  
 FAX: (408) 245-3108

Date: 03/10/97

Quoted To: AMDAHL CORPORATION  
 AMDAHL CORPORATION  
 POB 3470, ACCTS PAYABLE MS 112  
 Sunnyvale, CA 94088-3470

Quote No.: 6124  
 Customer No: 4  
 Phone No: (408) 746-6000  
 Page: 1

Salesperson: # 2- CASEY

Item Description	Qty	Unit Price	Amount	T
HIQ PENTIUM 133 MHZ SYSTEM	1	3999.00	3999.00	T
CPU INTEL PENTIUM 133MHZ	1	0.00	0.00	T
SPECIAL ORDER	1	0.00	0.00	T
P/I KP55T2P4 256K PIPELINE CACHE				
CPU FAN PC POWERCOOL PENTIUM	1	0.00	0.00	T
MEMORY 32MB 8*32 EDO 60NS	4	0.00	0.00	T
HDD WESTERN DIGITAL 2GB UW	1	0.00	0.00	T
CTRL ADAPTEC 2940UW PCI	1	0.00	0.00	T
FDD 1.44MB 3.5" FLOPPY DRIVE	1	0.00	0.00	T
VIDEO DIA S64 2500 2MB DRAM	1	0.00	0.00	T
CASE UL DESKTOP W. PS	1	0.00	0.00	T
CD-DR NEC 8X SCSI CDR-1410 OEM	1	0.00	0.00	T
ETHERNET INTEL PILA 8465B	2	0.00	0.00	T
KBO 104 KEYTRONIC SOFT W/ WIN	1	0.00	0.00	T
MICE MICROSOFT SERIAL 2.0	1	0.00	0.00	T
MONITOR 15" HITACHI NSA SUPERS	1	0.00	0.00	T
WARRANTY	0			
HIQ 3 YEAR AMDAHL WARRANTY				
3 YEARS ONSITE PARTS AND LABOR				
TPC BENCHMARK CLAUSE 7.3.4				
REPLACEMENT PARTS AVAILABLE FOR 5 YEARS				

John  
 737-5550  
 HOPEFULLY THAT LINE SEPERATE FOR THE  
 AMDAHL WARRANTY SATISFIES THE CLAUSE?  
 THANKS AGAIN,  
 CASEY

Sub-Total: 3999.00  
 Shipping: 0.00  
 Tax [ 7.75]: 309.92 \*  
 Quote Total: 4308.92

## Appendix E - Price Quotes

---



NETLUX

14180 Live Oak Ave., Unit E  
Baldwin Park, Ca. 91760

1-800-789-1780  
Phone #818-851-9737  
Fax #818-851-9837

March 27, 1997

Amdahl Corporation  
John Howell  
408-737-5550

### Quotation

---

Quantity	Description	Unit Price	Total
285	24-Port 10Base-T Rack Mount Ethernet Hub	\$251.00	\$71,535.00
		Total	\$71,535.00
		Shipping	TBD

Terms and Conditions:

FOB Origin  
5 Year Warranty  
Prices good for 60 Days  
24-Port will be scheduled for delivery over 1 month

Sincerely,  
Martin Parry  
NETLUX