TPC Benchmark™ C

Full Disclosure Report for

Siemens
Nixdorf

Informationssysteme AG

Primergy 560

Using Microsoft SQL Server 6.5 Enterprise Edition

and Microsoft Windows NT 4.0 Enterprise Edition

December 9, 1997

Second Edition

Second Edition December 9, 1997

Siemens Nixdorf Informationssysteme AG believes that the information in this document is accurate as of the publication date. The information in this document is subject to change without notice. We assume no responsibility for any errors that may appear in this document. The pricing information in this document is believed to accurately reflect the current prices as of the publication date. However, we provide no warranty of the pricing information in this document.

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

All performance data contained in this report were obtained in a rigorously controlled environment. Results obtained in other operating environments may vary significantly. We do 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.

Preface

The Transaction Processing Performance Council (TPC), of which Siemens Nixdorf Informationssysteme AG is a member, is an organization of computer companies, dedicated to the development of objective, industry-wide performance metrics in the area of transaction processing. Siemens Nixdorf Informationssysteme AG is involved in this effort, participating on the council and utilizing TPC benchmarks in performance evaluation.

The TPC Benchmark™ C Standard Specification was developed by the Transaction Processing Performance Council. This benchmark exercises the system components necessary to perform tasks associated with that class of on-line transaction processing (OLTP) environments emphasizing a mixture of read-only and update intensive transactions. This is a complex OLTP application environment exercising a breadth of system components associated by such environments characterized by:

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

This benchmark defines four on-line transactions and one deferred transaction, intended to emulate functions that are common to many OLTP applications. However, this benchmark does not reflect the entire range of OLTP requirements. The extent to which a customer can achieve the results reported by a vendor is highly dependent on how closely TPC-C approximates the customer application. The relative performance of systems derived from this benchmark does not necessarily hold for other workloads or environments. Extrapolations to any other environment are not recommended.

Benchmark results are highly dependent upon workload, specific application requirements, system 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.

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 subjected 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.

Summary

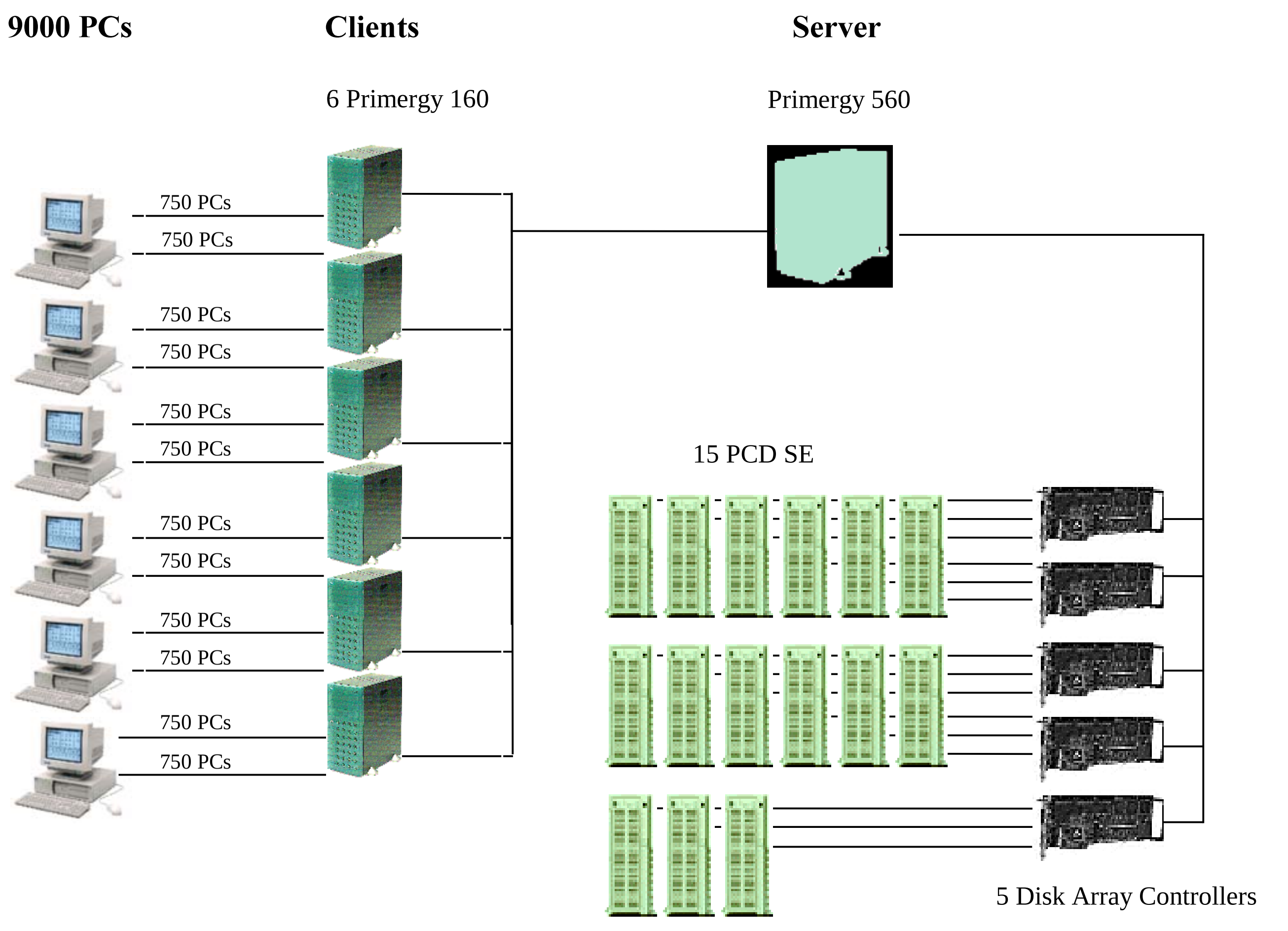
This report documents the TPC Benchmark™ C results achieved by the Siemens Nixdorf Informationssysteme AG using Microsoft SQL Server 6.5 Enterprise Edition.

The TPC Benchmark™ C tests were run on a Primergy 560 system using the Windows NT 4.0 Enterprise Edition operating system.

The results, summarized below, show the number of TPC Benchmark™ C transactions per minute (tpmC) and the price per tpmC ($/tpmC).

Software	**Hardware**	**tpmC**	**$/tpmC**
Microsoft SQL Server 6.5 Enterprise Edition, Windows NT 4.0 Enterprise Edition	Siemens Nixdorf Informationssysteme AG Primergy 560	10854.24	44.32$

SIEMENS NIXDORF Informationssysteme AG	Primergy 560 c/s with 6 Primergy 160		TPC-C REV 3.3.2 EXECUTIVE SUMMARY Page 1 of 2 Report Date: December 9, 1997	
Total System Cost	TPC-C Throughput	Price/Performance	Availability Date	
$ 481,008	10854.24 tpmC	$44.32/tpmC	January 1, 1998	
Processors	Database Manager	Operating-System	Other Software	Number of Users
4 Intel Pentium® Pro 200 MHz	Microsoft SQL Server 6.5 Enterprise Edition	Microsoft Windows NT 4.0 Enterprise Edition	Microsoft Internet Connector, Microsoft Visual C++, Microsoft SQL Server Programmer´s Toolkit, openUTM version 4.0 Transaction Monitor	9,000



System Components	Qty/Srv.	1 Primergy 560	Qty/Client	6 Primergy 160
Processors	4	Intel Pentium® Pro 200 MHz 1 MB SLC	1	Intel Pentium® Pro 200 MHz 256 KB SLC
Memory	3.25	GB	256	MB
Disk Controller	5	SCSI Controllers	1	SCSI Controller
Disk Drives	59	4 GB	1	2 GB
	53	9 GB		
Total GB of Storage	1	697.34 GB	1	2 GB

SIEMENS
NIXDORF

Informationssysteme AG

Primergy 560

Client/Server

TPC-C REV 3.3.2 EXECUTIVE SUMMARY
Page 2 of 2

Report Date: December 9, 1997

Description	Part Number	Third Party Brand	Pricing	Unit Price	Qty.	Extended Price	5 yr. Maint. Price
Server Hardware							
Base System	S26361-K412-V392			3 513 $	1	3 513 $	
PSU-Modul	S26113-E379-E10			179 $	2	359 $	
1. CPU Module	S26361-F1308-E1			455 $	1	455 $	
2. CPU Modul	S26361-F1308-E10			455 $	1	455 $	
Pentium Pro 200MHz/1MB	S26361-F1329-E202			5 747 $	4	22 989 $	
Kit for PPro 1MB SLC	S26361-F1718-E1			28 $	1	28 $	
Memory 1 GB (4x256) DIMM	S26361-F1307-E26			17 832 $	3	53 497 $	
Memory 256 MB (4x64) DIMM	S26361-F1307-E23			3 090 $	1	3 090 $	
Mylex Disc Array Controller PCI incl. 10% spare	S26361-F1779-E1			1 375 $	7	9 623 $	
Connectors for Disk Cabinets	S26361-F1222-E21			69 $	5	345 $	
Fast-Ether-Express-Pro 100Mbit (PCI)	S26361-F1465-E501			101 $	1	101 $	
Keyboard	S26381-K252-L165			39 $	1	39 $	
Country Pack	S26361-F1290-B173			37 $	1	37 $	
Sum Primergy 560							5 805 $
Monitor MCM 1405 ND	S26361-K449-V150			253 $	1	253 $	76 $
PCD-SE/W disk cabinet incl. 10% spare	S26361-K377-V291			1 655 $	17	28 138 $	
HD W-SCSI 4GB hot plug for PCD-Sew incl. 10% s	S26361-F1145-E140			943 $	65	61 264 $	
HD W-SCSI 9GB hot plug for PCD-Sew incl. 10% s	S26361-F1145-E181			1 664 $	59	98 198 $	
CD-ROM 8x for PCD-Sew	S26361-F726-E75			207 $	1	207 $	62 $
W-SCSI Cable UHD-HD incl. 10% spare	T26139-Y2549-V1			88 $	11	971 $	
W-SCSI Cable HD-HD incl. 10% spare	T26139-Y2527-M1			69 $	6	414 $	
2. Bridge Connector incl. 10% spare	S26361-F1148-L21			44 $	17	743 $	
					Subtotal	284 716 $	5 943 $
Server Software							
Microsoft NT-Server 4.0, Enterprise Edition	Microsoft	MS	1	3 999 $	1	3 999 $	
MS SQL-Server 6.5 Enterp.Edition unlim. license	Microsoft	MS	1	28 999 $	1	28 999 $	
					Subtotal	32 998 $	10 475 $
Client Hardware							
Primergy 160	S26361-K423-V744			3 191 $	6	19 145 $	
Keyboard	S26381-K252-V165			39 $	6	234 $	
Country Pack	S26361-F1454-B233			37 $	6	221 $	
Memory 64 MB EDO DIMM	S26361-F1514-E504			690 $	18	12 414 $	
Fast-Ether-Express-Pro 100Mbit (PCI)	S26361-F1465-E501			101 $	18	1 821 $	
Sum Primergy 160							23 520 $
Monitor MCM 1405 ND	S26361-K449-V150			253 $	6	1 517 $	455 $
					Subtotal	35 352 $	23 975 $
Client Software							
NT-Server 4.0		MS	1	809 $	6	4 854 $	
MS SQL-Server Prog. Toolkit		MS	1	499 $	1	499 $	
Open UTM	U11421-C10			973 $	6	5 838 $	8 820 $
MS Visual C++		MS	1	499 $	1	499 $	
					Subtotal	11 690 $	8 820 $
User Connectivity							
ATI 24 PORT HUB incl. 10% spare	AT-3024TR			160 $	413	66 080 $	
Fast Ethernet Hub 8*100 incl. 10% spare	AT-908TX-20			320 $	3	960 $	
					Subtotal	67 040 $	
					Total	431 796 $	49 213 $

Prices used in TPC bechmarks 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 section of the TPC benchmark pricing specifications. If you find that the stated prices are not available according to these terms, please inform the TPC at pricing@tpc.org. Thank you.

Five-Year Cost of Ownership: $481,008
tpmC Rating: 10854.24
$ / tpmC: 44.32

Note: The benchmark results and test methodology were audited by Francois Raab of Information Paradigm, Inc.

Numerical Quantities Summary

MQTh, computed Maximum Qualified Throughput	**10854.24 tpmC**
% throughput difference, reported & reproducibility runs	1.7 %

Response Times (in seconds)	90th percentile	Average	Maximum
- New-Order	1.71	0.92	9.59
- Payment	1.51	0.73	9.28
- Order-Status	2.45	1.39	9.70
- Delivery (interactive portion)	0.21	0.17	6.33
- Delivery (deferred portion)	4.96	2.45	14.40
- Stock-Level	3.75	2.12	11.88
- Menu	0.21	0.17	6.48

Transaction Mix, in percent of total transactions	
- New-Order	44.65 %
- Payment	43.07 %
- Order-Status	4.08 %
- Delivery	4.09 %
- Stock-Level	4.10 %

Emulation Delay (in seconds)	Response Time	Menu
- New-Order	0.1	0.1
- Payment	0.1	0.1
- Order-Status	0.1	0.1
- Delivery (interactive)	0.1	0.1
- Stock-Level	0.1	0.1

Keying/Think Times (in seconds)	Minimum	Average	Maximum
- New-Order	18.00/0.00	18.01/12.13	19.14/122.05
- Payment	3.00/0.00	3.01/12.12	4.02/122.02
- Order-Status	2.01/0.00	2.01/10.19	2.67/98.86
- Delivery (interactive)	2.01/0.00	2.01/ 5.11	2.99/ 45.48
- Stock-Level	2.01/0.00	2.01/ 5.18	2.88/ 52.01

Test Duration and Checkpointing	
- Ramp-up time	38 minutes
- Measurement interval	29 minutes
- Number of checkpoints	1
- Checkpoint interval	29 minutes
- Transactions during measurement interval (all types)	704955

Contents

Introduction

This is the Full Disclosure Report for the TPC Benchmark™ C running on the Siemens Nixdorf Informationssysteme AG system Primergy 560. It meets the requirements of the TPC Benchmark™ C Standard Revision 3.3.2.

Software and Hardware Configuration

This report documents the compliance of the Siemens Nixdorf Informationssysteme AG TPC Benchmark™ C tests using Microsoft SQL Server 6.5 Enterprise Edition Relational Database Management System.

The TPC Benchmark™ C tests were carried out on the Siemens Nixdorf Informationssysteme AG system Primergy 560. Primergy 560 is a powerful Windows NT Enterprise Server that features an Intel Pentium® Pro 200 MHz processors manufactured by Intel.

The processor power may be upgraded by three further Intel Pentium® Pro 200 MHz processors with highspeed onboard local memory access. The main memory capacity of the Primergy 560 scaled from 256 MB up to 4 GB. The Primergy system family uses the Windows NT 4.0 Enterprise Edition operating system.

Full Disclosure

From Clause 8.1 of the TPC Benchmark™ C Standard Specification:

> ...The intent of this disclosure is for a customer to be able to replicate the results of this benchmark given the appropriate documentation and products.

Siemens Nixdorf Informationssysteme AG believes that this full disclosure report meets the stated intention. Siemens Nixdorf Informationssysteme AG has strived to maintain the integrity of the Specification by adhering not only to the letter of the Specification, but also to its spirit.

Report Format

*The format of this document follows Clause 8 of the TPC Benchmark™ C specification (*TPC Benchmark™ C Standard Specification*, Revision 3.3.2, Transaction Processing Performance Council) which describes the full disclosure report requirements for the test.*

Each section of this report begins with the specification requirement printed in *italic type*. It is followed by plain type text that explains how the test complies with the requirement. Sections which require extensive listings reference appropriate appendices.

Report organization**:**

- General Items
- Clause 1 Related Items - Logical Database Design
- Clause 2 Related Items - Transaction and Terminal Profiles
- Clause 3 Related Items - Transaction and System Properties
- Clause 4 Related Items - Scaling and Database Population
- Clause 5 Related Items - Performance Metrics and Response Time
- Clause 6 Related Items - SUT, Driver, and Communication Definition
- Clause 7 Related Items - Pricing
- Clause 8 Related Items - Audit
- Appendix A - Application Source Code
- Appendix B - Database Details
- Appendix C - Tunable Parameters and Options
- Appendix D - Pricing Details
- Appendix E - Price Quotations
- Appendix F - Attestation Letter

Additional Copies

Additional copies of this report are available upon request from Siemens Nixdorf Informationssysteme AG:

Siemens Nixdorf Informationssysteme AG
Open Enterprise Computing
High End Server - Product Management

SNI OEC HES PM 4
Benchmarkcenter
Heinz-Nixdorf-Ring 1
33106 Paderborn
Germany

1. General Items

1.1 Application Code

The application program (as defined in Clause 2.1.7) must be disclosed. This includes, but is not limited to, the code implementing the five transactions and the terminal input and output functions. [Clause 8.1.1.4]

The source code of the application program is provided in Appendix A - Application Source Code.

1.2 Benchmark Sponsor

A statement identifying the benchmark sponsor(s) and other participating companies must be provided. [Clause 8.1.1.5]

This benchmark was sponsored and executed by Siemens Nixdorf Informationssysteme AG.
The benchmark was developed and engineered by Siemens Nixdorf Informationssysteme AG and Microsoft Corporation. Testing took place at SNI NT-benchmark laboratories in Paderborn, Germany.

1.3 Parameter Settings

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

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

[Clause 8.1.1.6]

The significant parameters and system configuration files are provided in Appendix C - Tunable Parameters and Options.

1.4 Configuration Diagrams

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

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

[Clause 8.1.1.7]

SUT Configuration

The Primergy 560 server system included:

4	Intel Pentium® Pro 200 MHz with 1 MB Second Level Cache
3.25	GB memory
5	SCSI controllers
86(59)	disks 4 GB measured configuration (priced configuration)
24(53)	disks 9 GB measured configuration (priced configuration)
1	LAN

The Primergy 160 client system included:

1	Intel Pentium® Pro 200 MHzwith 256 KB Second Level Cache
256	MB memory
1	SCSI controller
1	disk 2 GB
2 (3)	LAN measured configuration (priced configuration)

The benchmarked and priced system configurations are shown in Figure 1 and Figure 2 in accordance with Clause 8.1.1.7.

FIGURE 1: BENCHMARK SYSTEM CONFIGURATION PRIMERGY 560

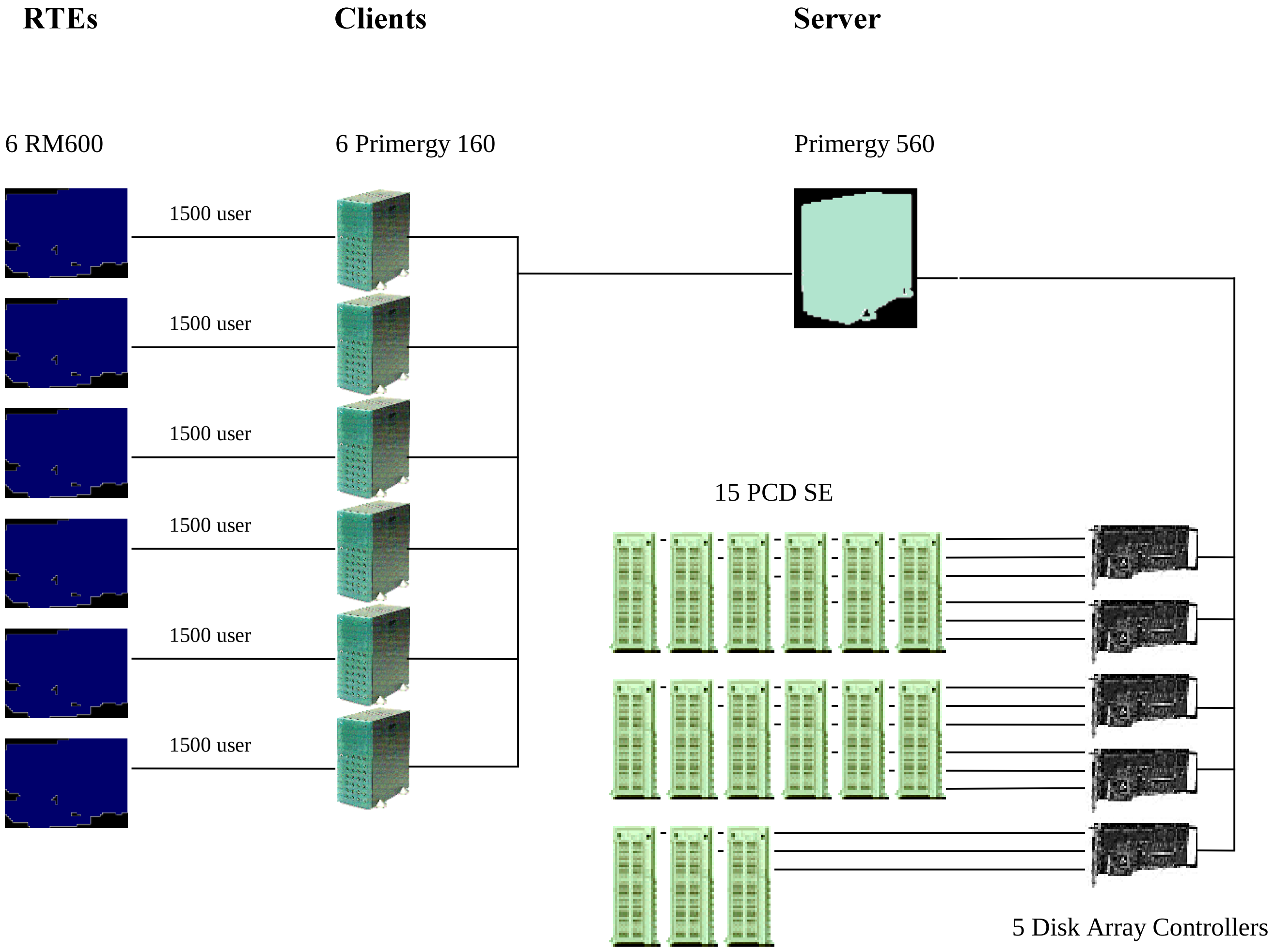
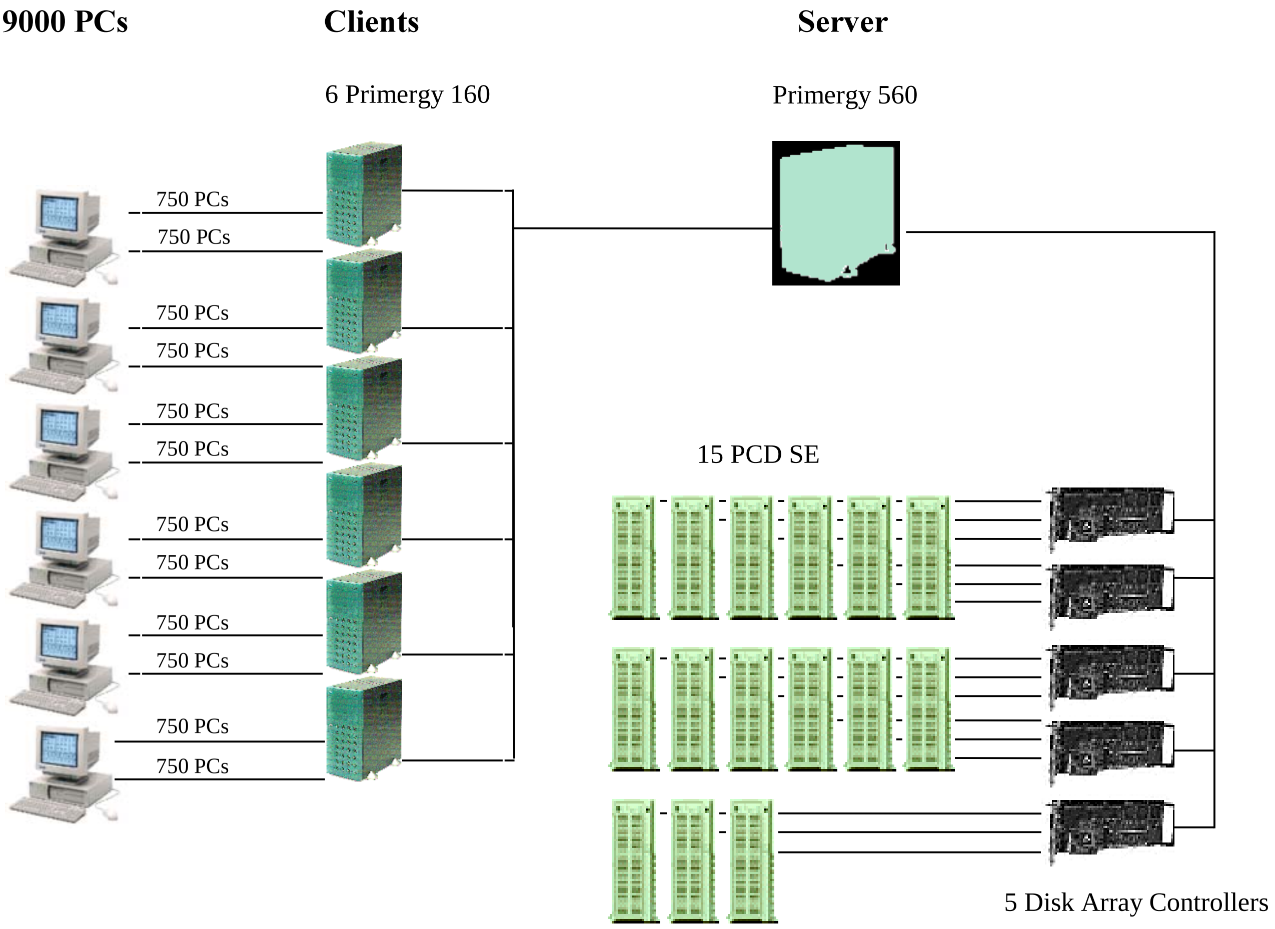


FIGURE 2: PRICED SYSTEM CONFIGURATION PRIMERGY 560



2. Clause 1 Related Items - Logical Database Design

2.1 Table Definitions

Listings must be provided for all table definition statements and all other statements used to set-up the database. [Clause 8.1.2.1]

The programs that defined, created, and populated the Microsoft SQL Server 6.5 Enterprise Edition database for this TPC benchmark™ C are listed in Appendix B - Database Details.

2.2 Physical Organization of Database

The physical organization of tables and indices, within the database, must be disclosed. [Clause 8.1.2.2]

FIGURE 1: PHYSICAL ORGANIZATION OF THE DATABASE

Mylex Controller 1					
Drive 0	1 x 4.3 GB	4 303 MB	System (unused)		C:
Drive 1	1 x 4.3 GB	4 303 MB	System		D:
Drive 2	21 x 4.3 GB	90 362 MB	data, spare, dump	RAID 0	F:, G:, Q:, V:

Mylex Controller 2					
Drive 0	21 x 4.3 GB	90 362 MB	data, spare, dump	RAID 0	H:, I:, R:, W:

Mylex Controller 3					
Drive 0	21 x 4.3 GB	90 362 MB	data, spare, dump	RAID 0	J:, K:, S:, X:

Mylex Controller 4					
Drive 0	21 x 4.3 GB	90 362 MB	data, spare, dump	RAID 0	M:, N:, T:, Y:

Mylex Controller 5					
Drive 0	16 x 8.7 GB	127 000 MB	data, spare	RAID 0	O:, P:
Drive 1	8 x 8.7 GB	34 732 MB	log	RAID 6	L:

Space was allocated to Microsoft SQL Server 6.5 Enterprise Edition on SUT disks according to the data in section 5.2. The size of the datafile on each disk drive was calculated to provide even distribution on load across the disk drives. The NT Disk Administrator was used to create raw devices for data and NTFS partitions for dump and log devices. For further information see Appendix B (Disk Usage) and Figure 1 in 5.2 (Distribution of Tables and Log). No attempt was made to alter the default physical organization of the database tables and indices chosen by Microsoft SQL Server 6.5 Enterprise Edition.

2.3 Insert and Delete Operations

It must be ascertained that insert and/or delete operations to any of the tables can occur concurrently with the TPC-C transaction mix. Furthermore, any restriction in the SUT database implementation that precludes inserts beyond the limits defined in Clause 1.4.11 must be disclosed. This includes the maximum number of rows that can be inserted and the maximum key value for these new rows. [Clause 8.1.2.3]

There were no restrictions on insert and delete operations to any tables.

2.4 Database Partitioning

While there are a few restrictions placed upon horizontal or vertical partitioning of tables and rows in the TPC benchmark™ C (see Clause 1.6), any such partitioning must be disclosed. [Clause 8.1.2.4]

There was no partitioning used in this implementation.

2.5 Replication of Tables

Replication of tables, if used, must be disclosed (see Clause 1.4.6). [Clause 8.1.2.5]

Replication of tables was not used in this implementation.

2.6 Additional and/or Duplicated 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 and/or duplicated attributes were used.

3. Clause 2 Related Items - Transaction and Terminal Profiles

3.1 Random Number Generator

The method of verification for the random number generation must be described. [Clause 8.1.3.1]

The driver code of the RM 600 RTE generates random numbers by using three C-library routines lrand48(), srand48() and drand48(), available in RELIANT-UNIX ®.

lrand48() is a member of the family of functions which generate pseudo-random numbers using the well-known linear congruential algorithm and 48-bit integer arithmetic.

The function lrand48() returns non-negative long integers uniformly distributed over the interval $[0, 2^{31}-1]$. It works by generating a sequence of 48-bit integer values, X_n, according to the linear congruential formula

$$X_{n+1}=(aX_n+c) \bmod m; \quad n>0.$$

The parameter m is 2^{48}; hence 48-bit integer arithmetic is performed.

The value returned by the function lrand48() is computed by first generating the next 48-bit X_n in the sequence. Then the appropriate number of bits, according to the type of data item to be returned, are copied from the high-order (leftmost) bits of X_n and transformed into the returned value.

3.2 Input/Output Screen Layout

The actual layouts of the terminal input/output screens must be disclosed. [Clause 8.1.3.2]

The screen layout corresponded exactly to those of the TPC-C Standard Specification (specified in Clause 2.4.3, 2.5.3, 2.6.3, 2.7.3, and 2.8.3).

3.3 Configured Terminal Features

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

The Primergy 160 is commercially available. All of the requirements in clause 2.2.2.4. are supported. This was verified by manually exercising each specification on a Primergy 160.

3.4 Presentation Managers or Intelligent Terminals

Any usage of presentation managers or intelligent terminals must be explained. [Clause 8.1.3.4]

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

3.5 Transaction Statistics

The numerical quantities which are required are listed in the following table. [Clause 8.1.3.5 to 8.1.3.11]

	Statistics	Percentage
New-Order	Home order-lines	98.99%
	Remote order-lines	1.01%
	Rolled back transactions	1.00%
	Average items per order	10.00
Payment	Home transactions	85.11%
	Remote transactions	14.89%
	Non-primary key access	60.20%
Order-Status	Non-primary key access	60.32
Delivery	Skipped transactions	0
Transaction Mix	New-Order	44.65 %
	Payment	43.07 %
	Order-Status	4.08 %
	Delivery	4.09 %
	Stock-Level	4.10 %

3.6 Queueing Mechanism

The queuing mechanism used to defer the execution of the Delivery transaction must be disclosed. [Clause 8.1.12]

The client application processes submitted delivery transactions to named pipe delivery server software running on the client machines There was a single delivery server 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 - Application Source Code.

4. Clause 3 Related Items - Transaction and System Properties

ACID Tests

The results of the ACID tests must 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 tests were performed successfully. The following sections describe the requirements of each of the tests as described in Clause 3 and the approach used to satisfy them.

All ACID tests were performed on the Primergy 560 system using the fully scaled database, except for the test of durable media failure.
The durability test was performed on a database scaled to 10 warehouses. This test would also pass on a fully scaled database.

4.1 Atomicity

The system under test must guarantee that database transactions are atomic; the system will either perform all individual operations on the data, or will assure that no partially-completed operations leave any effects on the data. [Clause 3.2.1]

Commit Transaction

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) and verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have been changed appropriately. [Clause 3.2.2.1]

The following steps demonstrated atomicity for completed (COMMIT) transactions:

- A row was randomly selected from the warehouse, district and customer table.
- the current balance was noted.
- A payment transaction was executed with the above identifiers and a known amount.
- The transaction was committed.
- It was verified, that the rows contain the correct updated balances.

Rollback Transaction

Perform the Payment transaction for a randomly selected warehouse, district, and customer (by customer number as specified in Clause 2.5.1.2) and substitute a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the records in the CUSTOMER, DISTRICT, and WAREHOUSE tables have NOT been changed. [Clause 3.2.2.2]

The following steps demonstrated atomicity for aborted (ROLLBACK) transactions:

- A row was randomly selected from the warehouse, district and customer table.
- the current balance was noted.
- A payment transaction was executed with the above identifiers and a known amount.
- The transaction was rolled back.
- It was verified, that the rows contain the original balances.

4.2 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. [Clause 3.3.1]

Consistency conditions 1 - 4 were tested by issuing queries to the database. The results of the queries verified that the database was consistent for all these tests. The tests were performed before and after the performance run on the same database that was used for the benchmark.

4.3 Isolation

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

We ran all of the seven isolation tests as described in clause 3.4.2.1 to 3.4.2.7 and additionally the two phantom protection tests. The tests were executed using shell scripts to issue queries to the database. The results of the queries verified that the required isolation had been met.

4.4 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. [Clause 3.5]

List of single failures:

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

[Clause 3.5.3]

The intent of these tests is to demonstrate that all transactions whose output messages have been received at the terminal or RTE have in fact been committed in spite of any single failure from the list in Clause 3.5.3 and that all consistency conditions are still met after the database is recovered.

It is required that the system crash test(s) and the loss of memory test(s) described in Clause 3.5.3.2 and 3.5.3.3 be performed under full terminal load and a fully scaled database. The durable media failure test(s) described in Clause 3.5.3.1 may be performed on a subset of the SUT configuration and database. For the SUT subset, all multiple hardware components, such as processors and disk / controllers in the full SUT configuration, must be represented by the greater of 10% of the configuration or two of each of the multiple hardware components. The database must be scaled to at least 10% of the fully scaled database, with a minimum of two warehouses. ... Furthermore, the standard driving mechanism must be used in this test. The test sponsor must state that to the best of their knowledge, a fully scaled test would also pass all durability tests. [Clause 3.5.4]

The failure of all or part of memory test and the system crash test were combined with the loss of log disk and performed under full load and by using a fully scaled database.

In accordance with Clause 3.5.4, the full Hardware configuration of the SUT was used during the all durability test, except the test for loss of data.

The durable media failure test for loss of data disk was performed with 29 of the 109 disks and a database scaled to 10 warehouses under the load of 100 users. To the best of the test sponsor's knowledge, a fully loaded and fully scaled database would also pass this durability test.

All durability tests used the following procedure:

- The database was backed up.
- The current count of the total number of orders was determined by summing up the D_NEXT_O_ID fields of all rows in the DISTRICT table before the test.
- A 15 minutes test was run.
- One or more failures were induced (see the list below).
- The steps necessary to perform recovery were performed.
- The current count of the total number of orders was determined and compared with the pre-test result to show that the count changed by the amounts of the completed New-Order transactions.
- the database was sampled for orders from the success file.

Three separate failures were induced on the SUT to demonstrate compliance with the durability requirements stated in Clause 3.5.

A. Irrecoverable loss of a Logical Log device (Clause 3.5.3.1).
   This failure was induced by pulling up the disk drive containing transaction log data in order to 'force' an I/O error. This was noticed on the very next access to one member of the mirrored pair set containing the transaction log. Since mirroring is handled transparently no database recovery was required. The durability drivers continued to run unaffected.

B. Irrecoverable loss of a disk that contained database tables (Clause 3.5.3.1).
   Prior to these tests, a backup of the database was made. The failure was induced by pulling up a single disk that stored user data.
   To recover SQL Server was shut down. After restart the transaction log was dumped, the disk was replaced on the SUT, the backup was restored, and the database was rolled forward based on information of the transaction dump.

C. Instantaneous interruption of system power requiring system reboot to recovery (Clause 3.5.3.2).
   Because power failure destroys the contents of the memory, recovery from power failure also meets the memory failure requirement stated in Clause 3.5.3.3. This failure was induced by turning off the power to the SUT 5 minutes after media failure test for loss of log.
   To recover from this failure, the power was restored to the SUT and SQL Server was restarted. SQL Server used the logical logs to automatically roll forward all committed transactions and roll back uncommitted changes.

5. Clause 4 Related Items - Scaling and Database Population

5.1 Initial Cardinality of Tables

The cardinality (e.g., the number of rows) of each table, as it existed at the start of the benchmark run (see Clause 4.2), must be disclosed. If the database was over-scaled and inactive rows of the WAREHOUSE table were deleted (see Clause 4.2.2), the cardinality of the WAREHOUSE table as initially configured and the number of rows deleted must be disclosed. [Clause 8.1.5.1]

The database for the Primergy 560 system was scaled for 900 warehouses. 10 rows of the WAREHOUSE table were deleted. In accordance with Clause 4.2, the following number of records were loaded in the specified tables:

Table	Number of Records
Warehouse	900
District	9,000
Customer	27,000,000
History	27,000,000
Order	27,000,000
New-Order	8,100,000
Order-Line	269,999,044
Stock	90,000,000
Item	100,000

The following constant values were used during the database build and benchmark test for the NURand function:

Constant C	Value
C_LAST (build)	123
C_LAST (run)	223
C_ID	999
OL_ID	23

5.2 Distribution of Tables and Log

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

FIGURE 1: LOGICAL ORGANIZATION OF THE DATABASE

	device	raw size	use
L:	tpclog1	22 000 MB	Log
F:	tpcmisc1	1 200 MB	Warehouse,District,Item, New Order,History,Order
H:	tpcmisc2	1 200 MB	Warehouse,District,Item, New Order,History,Order
J:	tpcmisc3	1 200 MB	Warehouse,District,Item, New Order,History,Order
M:	tpcmisc4	1 200 MB	Warehouse,District,Item, New Order,History,Order
O:	tpcmisc5	1 200 MB	Warehouse,District,Item, New Order,History,Order
Q:	tpcol1	7 000 MB	Orderline
R:	tpcol2	7 000 MB	Orderline
S:	tpcol3	7 000 MB	Orderline
T:	tpcol4	7 000 MB	Orderline
G:	tpcsc1	12 000 MB	Stock,Customer
I:	tpcsc2	12 000 MB	Stock,Customer
K:	tpcsc3	12 000 MB	Stock,Customer
N:	tpcsc4	12 000 MB	Stock,Customer
P:	tpcsc5	12 000 MB	Stock,Customer

5.3 Database Model, Interface, and Access Language

A statement must be provided that describes:

1. The data model implemented by the DBMS used (e.g., relational, network, hierarchical)

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

[Clause 8.1.5.3]

Microsoft SQL Server 6.5 Enterprise Edition is a Relational DataBase Management System. The interface used was Microsoft SQL Server 6.5 Enterprise Edition stored procedures accessed with Remote Procedure Calls embedded in C code.

5.4 Database Partitions/Replications Mapping

The mapping of database partitions/replications must be explicitly described. [Clause 8.1.5.4]

There was no partitioning and/or replication used in this implementation.

5.5 180 day space Calculation

Details of the 180-day space computations along with proof that the database is configured to subtain 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]

Calculations of space requirements in the priced configurations for the 180-day period are provided in Appendix D - Pricing Details.

6. Clause 5 Related Items - Performance Metrics and Response Time

6.1 Measured tpmC

Measured tpmC must be reported. [Clause 8.1.6.1]

During the 29 minutes measurement period on the Primergy 560 the throughput measured was 10854.24 tpmC.

6.2 Response Times

Ninetieth percentile, maximum and average response times must be reported for all transaction types as well as for the Menu response time. [Clause 8.1.6.2]

Type	Average	Maximum	90 Percentile
New-Order	0.92	9.59	1.71
Payment	0.73	9.28	1.51
Order-Status	1.39	9.70	2.45
Interactive Delivery	0.17	6.33	0.21
Deferred Delivery	2.45	14.40	4.96
Stock-Level	2.12	11.88	3.75
Menu	0.17	6.48	0.21

6.3 Keying and Think Times

The minimum, the average, and the maximum keying and think times must be reported for each transaction type. [Clause 8.1.6.3]

Keying Times			
Type	**Average**	**Maximum**	**Minimum**
New-Order	18.01	19.14	18.00
Payment	3.01	4.02	3.00
Order-Status	2.01	2.67	2.01
Delivery	2.01	2.99	2.01
Stock-Level	2.01	2.88	2.01

Think Times			
Type	**Average**	**Maximum**	**Minimum**
New-Order	12.13	122.05	0.00
Payment	12.12	122.02	0.00
Order-Status	10.19	98.86	0.00
Delivery	5.11	45.48	0.00
Stock-Level	5.18	52.01	0.00

6.4 Graphs

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

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]

Think Time frequency distribution curves (see Clause 5.6.3) must be reported for each transaction type. [Clause 8.1.6.6]

Keying Time frequency distribution curves (see Clause 5.6.4) must be reported for each transaction type. [Clause 8.1.6.7]

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 1: NEW-ORDER RESPONSE TIME DISTRIBUTION

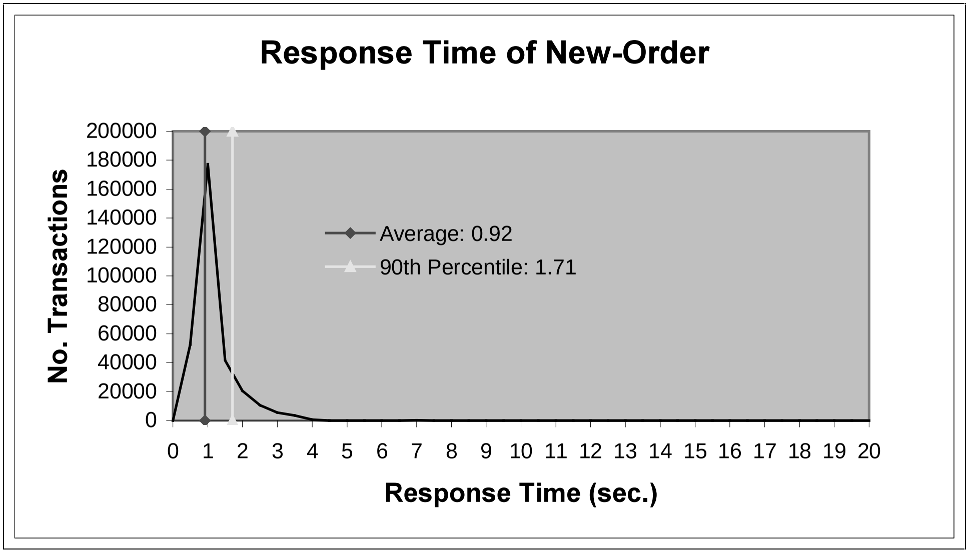


FIGURE 2: PAYMENT RESPONSE TIME DISTRIBUTION

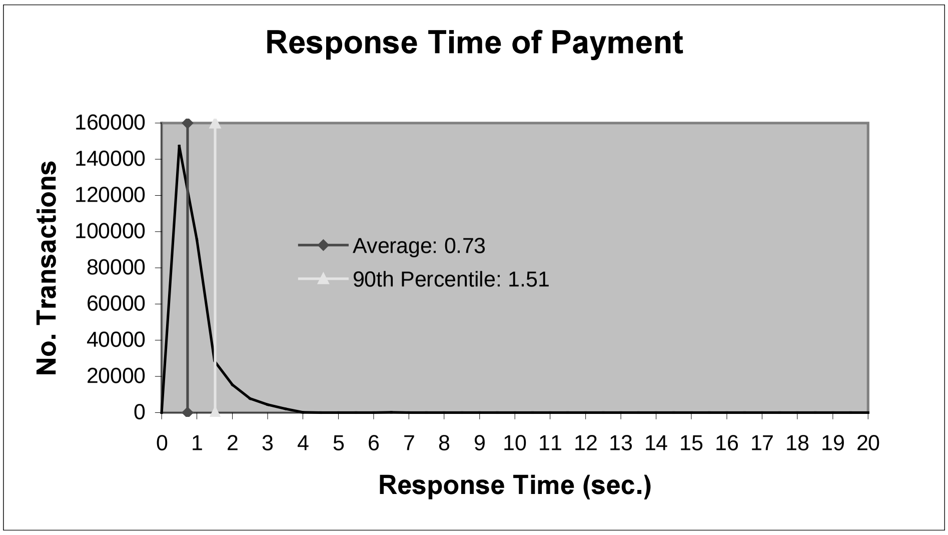


FIGURE 3: ORDER-STATUS RESPONSE TIME DISTRIBUTION

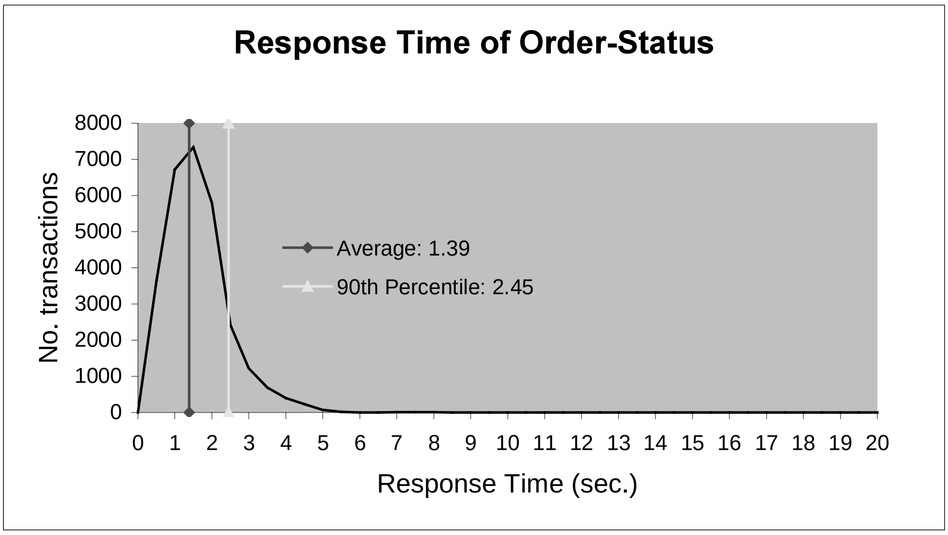


FIGURE 4: DELIVERY RESPONSE TIME DISTRIBUTION

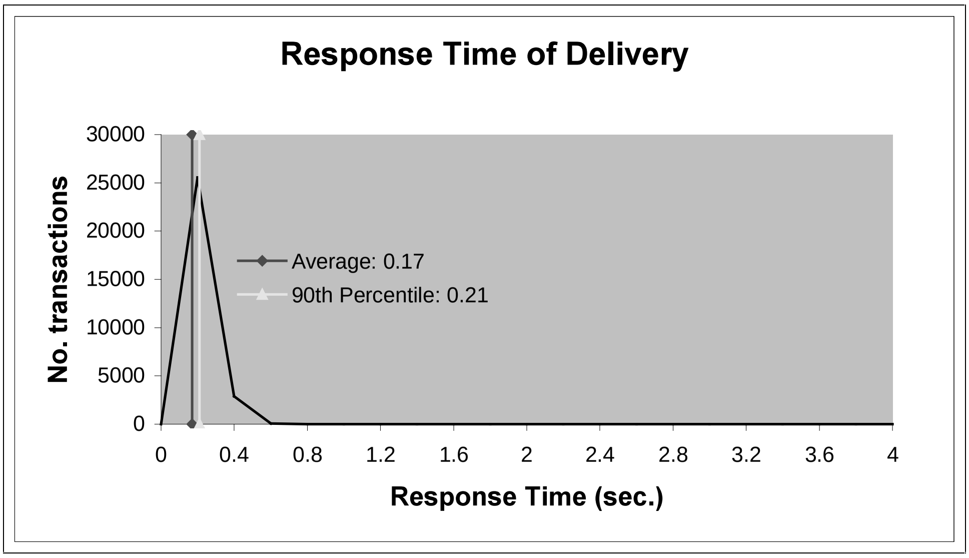


FIGURE 5: STOCK-LEVEL RESPONSE TIME DISTRIBUTION

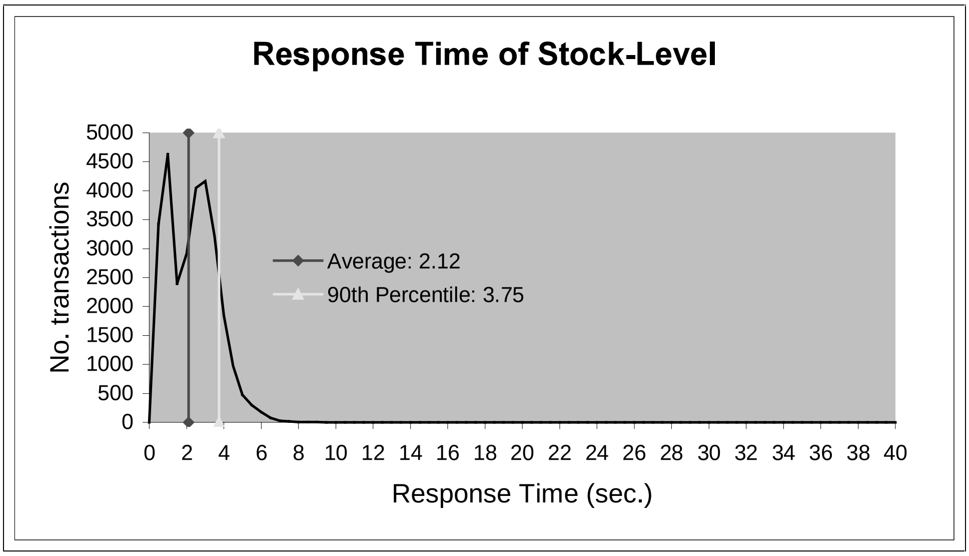


FIGURE 1: RESPONSE TIME VERSUS THROUGHPUT

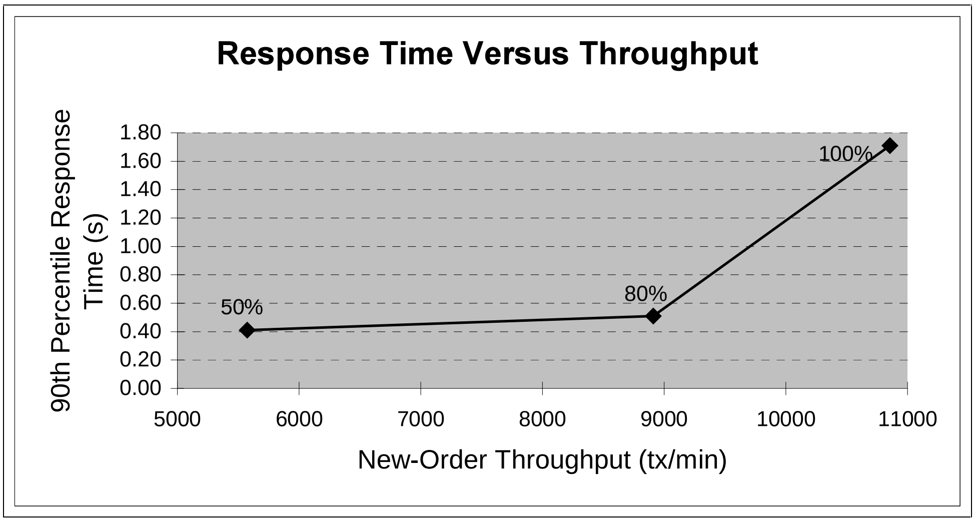


FIGURE 1: NEW-ORDER THINK TIME DISTRIBUTION

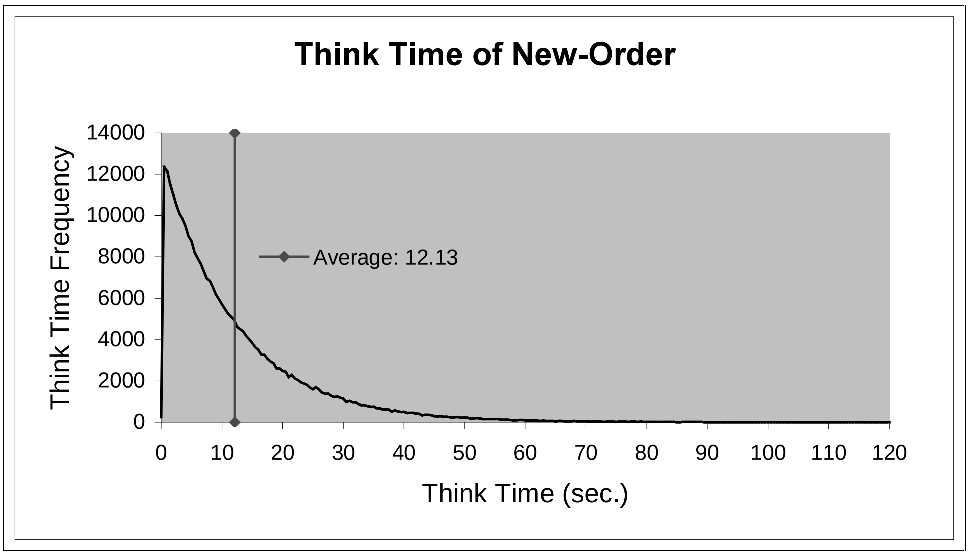
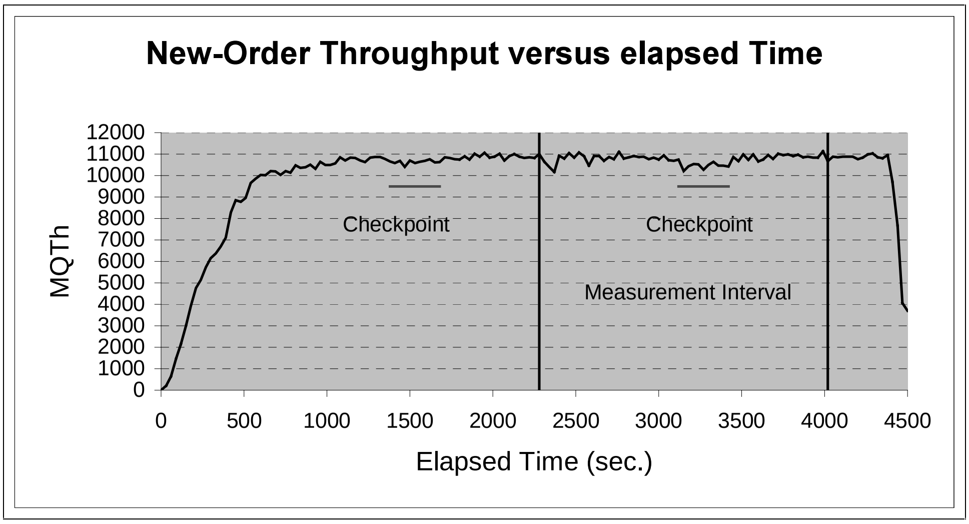


FIGURE 1: THROUGHPUT VERSUS ELAPSED TIME



6.5 Steady State Determination

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]

In all test runs, steady state was achieved before the measurement period began. Steady state was determined to occur based on a visual inspection of tpmC versus time (see graph in section 6.4).

The graph in section 6.4 illustrates that the measurement period was within the steady state period for the run. One checkpoint occurred during the measurement period.

6.6 Work Performed

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 and captured in RTE log files before being transmitted. There was one log file for each user. The input screen for the requested transaction was returned and it was also captured and timestamped in the RTE log files. The difference between these two timestamps was the menu response time.

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 and captured in RTE log files. The return of the screen with the required response data was timestamped and captured in the RTE log files. The difference between these two timestamps was the response time for that transaction.

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

The RTE transmissions were sent to Internet Information Server running on the client machines through Ethernet LANs. Internet Information Server handled all screen I/O as well as all requests to the database on the server. Internet Information Server communicated with the database server over openUTM which was used as transaction monitor. The frontend program (openUTM client) handled all incoming requests on the client system while the backend program (openUTM server) forwarded all requests to the database on the server system. The front-end programs communicated with the back-end programs through openUTM calls. openUTM routes the transaction and balances the load according to the options defined in the openUTM configuration file listed in Appendix C.

All database operations like update, select, delete and insert are performed by one of the TPC-C back end programs. The TPC-C backend program commits the transaction after all the corresponding operations are done.

Modified database buffers are migrated to disk a least-recently-used basis independent of transaction commits. In addition, every block modification is protected by log records. Asynchronously the log buffers are flushed to a log file on disk either when the transaction is committed or when the log buffer’s fill state reaches it’s limit. The log buffer’s always flushed by a commit before it become full.

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 29 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.

6.7 Reproducibility

A description of the method used to determine the reproducibility of the measurement results must be reported. [Clause 8.1.6.11]

The Primergy 560 system test was run twice to ensure the reproducibility of the results. The reproducibility test run under exactly the same conditions as the reported test. All tests conform to the TPC rules.

The tpmC result from the reproducibility test was within 1.7% of the reported tpmC.

In the following, both results are shown to document the reproducibility:

	tpmC
reported test	10854.24
reproducibility test	10669.31

6.8 Duration of Measurement

A statement of the duration of the measurement interval for the reported Maximum Qualified Throughput (tpmC) must be included. [Clause 8.1.6.12]

The measurement interval of the Primergy 560 system test was 29 minutes. This measurement interval corresponds to the amount of time from the beginning of one checkpoint to the beginning of the next (which, actually, is less than the amount of time it takes to fill a log file).

6.9 Regulation of Transaction Mix

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

The transaction mix was regulated by weighted random distribution. The chosen weights meet the required minimum percentages of the mix which are described in Clause 5.2.3 of the Standard Specifications. During the measurement interval the weights were controlled and if necessary adjusted by the RTE. The adjustments did not exceed the allowed limit of 5%.

6.10 Transaction Mix

The percentage of the total mix for each transaction type must be disclosed. [Clause 8.1.6.14]

	Percentage
New-Order	44.65 %
Payment	43.07 %
Order-Status	4.08 %
Delivery	4.09 %
Stock-Level	4.10 %

6.11 Transaction Statistics

The percentage of New-Order transactions rolled back as a result of invalid item number must be disclosed. [Clause 8.1.6.15]

The average number of order-lines entered per New-Order transaction must be disclosed. [Clause 8.1.6.16]

The percentage of remote order-lines entered per New-Order transaction must be disclosed. [Clause 8.1.6.17]

The percentage of remote Payment transactions must be disclosed. [Clause 8.1.6.18]

The percentage of customer selections by customer last name in the Payment and Order-Status transactions must be disclosed. [Clause 8.1.6.19]

The percentage of Delivery transactions skipped due to there being fewer than necessary orders in the New-Order table must be disclosed. [Clause 8.1.6.20]

The numerical quantities which are required in Clause 8.1.6.15 to 8.1.6.20 are already listed in a table above (see section 3.5).

6.12 Checkpoint Statistics

The number of checkpoints in the Measurement Interval, the time in seconds from the start of the Measurement Interval to the first checkpoint and the Checkpoint Interval must be disclosed. [Clause 8.1.6.21]

There was one checkpoint before and one during the measurement interval. The second checkpoint occurred 900 seconds after the start of the measurement interval. The checkpoint interval was set to 1740 seconds.

7. Clause 6 Related Items - SUT, Driver, and Communication Definition

7.1 RTE Inputs

If the RTE is commercially available, then its inputs must be specified. Otherwise, a description must be supplied of what inputs (e.g., scripts) to the RTE had been used. [Clause 8.1.7.1]

The driver used for the TPC Benchmark™ C test is a propietary driver.

The proprietary driver resided on an external Driver System and performed the following functions during the benchmark:

- Emulates a user entering input data on a Web-Browser by generating and sending HTML-Pages to the SUT,
- Emulates a Web-Browser displaying output messages by receiving response messages from the SUT,
- Emulates a Web-Browser delay time,
- Records response times,
- Performs conversion and/or multiplexing into the communications protocol used by the communications interface between the driver and the SUT, and
- Performs statistical accounting.

The proprietary driver performs only those functions stated in Clause 6.4.2. The driver does not perform any special functions to enhance the performance.

7.2 Functionality and Performance of Emulated Components

It must be demonstrated that the functionality and performance of the components being emulated in the Driver System are equivalent to that of the priced system. The results of the test described in Clause 6.6.3.4 must be disclosed. [Clause 8.1.7.2]

The Driver System consisted of a RM600 Model 420. This driver was attached to the client machine through an Ethernet LAN. Since this is exactly the same connectivity as configured in the priced system, no component was emulated. Therefore, the test described in Clause 6.6.3.4 was not required.

7.3 Functional Diagrams of the Benchmarked and Proposed Configuration

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]

Figure 1 and Figure 2 in section 1.4 show the functional diagrams of the benchmark configuration and the priced configuration.

7.4 Network Configurations of the Tested and Proposed Services

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]

Figure 1 and Figure 2 in section 1.4 also show the network setup of both configurations. The driver replaces the workstations.

In the tested configuration one standard ethernet LAN segments was used to connect the server with the clients and six standard ethernet LAN segments were used to connect the clients with the six RTE systems.

In the priced configuration twelve standard ethernet LAN segments were used, each to connect 750 workstations with one client.

7.5 Network Bandwidth

The bandwidth of the network(s) used in the tested / priced configuration must be disclosed. [Clause 8.1.7.5]

The Ethernet used in the local area network (LAN) between the emulated user system and the front-end system complies with the IEEE 802.3 standard and it’s bandwith is 100 Mbps.

7.6 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]

The Primergy 560 requires no operator intervention to sustain the reported throughput.

8. Clause 7 Related Items - Pricing

8.1 System Pricing

A detailed list of hardware and software used in the priced system must be reported. Each separately orderable item must have vendor part number, description, and release/revision level, and either general availability status or committed delivery date. If package-pricing is used, vendor part number of the package and a description uniquely identifying each of the components of the package must be disclosed. Pricing source(s) and effective date(s) of price(s) must also be reported. [Clause 8.1.8.1]

The total 5-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is recommended. The basis of all discounts used must be disclosed. [Clause 8..1.8.2]

The details of the hardware and software are reported in the summary in front of this report. The spreadsheet used to determine the 5-year price and the speadsheet used to describe the priced configuration can be found in Appendix D - Pricing Details.

8.2 Availability Dates

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, the reported availability date for the priced system must be the date at which all components are committed to be available. [Clause 8.1.8.3]

All hardware and software components used in the price calculations of the Primergy 560 system will be generally available from Siemens Nixdorf Informationssysteme AG as of January 1, 1998.

8.3 Throughput and Price/Performance

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

Primergy 560 system was measured at 10854.24 tpmC with Microsoft SQL Server 6.5 Enterprise Edition with a 5-year system price of $481,008. The respective price/performance for the Primergy 560 is $44.32/tpmC. The priced Primergy 560 will be available as of January 1, 1998.

8.4 Country Specific Pricing

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

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

8.5 Usage Pricing

For any usage pricing, the sponsor must diclose:

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

[Clause 8.1.8.6]

The component pricing based on usage is shown below:

- One Microsoft Windows NT Server 4.0 license (includes 5 client access licenses)
- One Microsoft Windows NT Server, Enterprise Edition 4.0 license (includes 25 client access licenses)
- One Microsoft SQL Server, Enterprice Edition 6.5 license (includes unlimited user license)
- One Microsoft SQL Workstation (includes programmers toolkit)
- Microsoft Visual C++ 32-bit edition

9. Clause 8 Related Items - Audit

> The auditor’s name, address, phone number, and a copy of the auditor’s attestation letter indicating compliance must be included in the Full Disclosure Report.
>
> A review of the pricing model is required to ensure that all components required are priced (see Clause 9.2.8). The auditor is not required to review the final Full Disclosure Report or the final pricing prior to issuing the attestation letter. [Clause 8.1.9]

The benchmark test of the Primergy 560 system with Microsoft SQL Server 6.5 Enterprise Edition was independently audited by:

Francois Raab, a TPC certified auditor with Information Paradigm, Inc. of Colorado Springs, CO.

The attestation letter is included in Appendix F.

Requests for this TPC-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

SNI OEC HES PM 4
Benchmarkcenter
Heinz-Nixdorf-Ring 1
33106 Paderborn
Germany

Appendix A - Application Source Code

Include Files

```
/*      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 mesage 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);
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);


#ifndef ERROR_H_INCLUDED
#define ERROR_H_INCLUDED
// extern TERM Term;
// 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;
void WriteZString( EXTENSION_CONTROL_BLOCK *pECB, char *szStr);
void WINAPI ErrorMessage( EXTENSION_CONTROL_BLOCK *pECB, int iError,
int iErrorType, char *szMsg, int iTermId, int iSyncId);

#define ERR_BAD_ITEM_ID                     1               // expected
abort record in txnRecord
#define ERR_TYPE_DELIVERY_POST        2               // expected
delivery post failed
```

```
#define ERR_TYPE_WEBDLL                       3               // tpcc web
generated error
#define ERR_TYPE_SQL                  4               // sql server
generated error
#define ERR_TYPE_DBLIB                5               // dblib generated
error
#define ERR_TYPE_ODBC                 6               // odbc generated
error
#define ERR_TYPE_SOCKET                       7               // error on
communication socket client rte only
#define ERR_TYPE_DEADLOCK             8               // dblib and odbc
only deadlock condition
#define ERR_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 corrisponding Item_Id.
#define ERR_NEWORDER_QTY_WITHOUT_ITEMID               1040 //" New Order
Qty entered without a corrisponding 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 Last Name 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*".
#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*".
#endif
```

```
/********
*
* 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;
//
// 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_

#ifndef PIPE_ROUTINES_H_INCLUDED
#define PIPE_ROUTINES_H_INCLUDED

#ifdef _DEBUG
__inline void __cdecl Trace(PSTR pFormat, ...)
{
 va_list Parameter ;

 va_start(Parameter, pFormat) ;

 vfprintf(stderr, pFormat, Parameter) ;
}
#else
__inline void __cdecl Trace(PSTR pFormat, ...) {}
#endif

#define UTM_MEM_SPACE  "SniUtmPipeMem"
#define UTM_MEM_EVENT  "SniUtmEvent"

typedef struct
{
 HANDLE  evIisReq ;
 HANDLE  evUtmAck ;
 HANDLE   hThread ;           // Handle of the UTM-Service-Thread ;
 DWORD   dwProId ;            // Id of process who handles the IIS-
Requests
} UTM_HANDLES ;


typedef struct
{
 DWORD  dwMaxConnections ;       // Max. Connections
 long    lConnections ;          // Current Connections
 DWORD  dwCpp ;                  // Connections per Process

 DWORD  dwMaxTransferLen ;       // Size for the transfer buffer IIS <-
-> UTM-Client

 DWORD  dwPIdMasterUtm ;         // Process Id from the first (Master-)
UTM-Client
 HANDLE evTerminate ;
 HANDLE smBreak ;

 UTM_HANDLES UtmHandles[] ;
} UTM_SHARED_MEM ;


typedef struct
{
 HANDLE  evRDav ;                 // RDav = Read data available  (UTM-
View)
 HANDLE  evWDav ;                 // WDav = Write data available (UTM-
View)
 HANDLE  hStop ;                  // Stop received

 DWORD   dwMaxTransferLen ;

 LPBYTE  lpBuffer ;
 LPDWORD lpLen ;
} SM_PIPE ;


HANDLE DuplicateUtmHandle(HANDLE hSrc, DWORD dwProId) ;
BOOL OpenClientPipe(SM_PIPE *pPipe, DWORD dwId, UTM_SHARED_MEM
*lpUtmMem) ;
BOOL OpenServerPipe(SM_PIPE *pPipe, DWORD dwId, LPSECURITY_ATTRIBUTES
lpEventAttributes, UTM_SHARED_MEM *lpUtmMem) ;
BOOL ReadPipe(SM_PIPE *pPipe, void *Buffer, DWORD BufSize, DWORD
*pnRead) ;
BOOL WritePipe(SM_PIPE *pPipe,void *Buffer, DWORD BytesToWrite, DWORD
*pnWritten) ;

#endif

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

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

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

```
BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId,
                                             DBPROCESS **dbproc, char
*server, char *database,
                                             char *user, char
*password, char *app, int *spid);
BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS
*dbproc);
BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId,
                                             DBPROCESS *dbproc,
STOCK_LEVEL_DATA *pStockLevel,
                                             short deadlock_retry);
int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId,
                                             DBPROCESS *dbproc,
NEW_ORDER_DATA *pNewOrder,
                                             short deadlock_retry);
int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId,
                                             DBPROCESS *dbproc,
PAYMENT_DATA *pPayment,
                                             short deadlock_retry);
int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId,
                                             DBPROCESS *dbproc,
ORDER_STATUS_DATA *pOrderStatus,
                                             short deadlock_retry);
BOOL SQLInit(void);
void SQLCleanup(void);
BOOL SQLThreadAttach(void);
BOOL SQLThreadDetach(void);
PECBINFO SQLGetECB(PDBPROCESS p);

#ifndef TPCC_H_INCLUDED
#define TPCC_H_INCLUDED
extern char szErrorLogPath[];

#ifdef UTM_SERVER
typedef char EXTENSION_CONTROL_BLOCK;
extern EXTENSION_CONTROL_BLOCK *gpECB;
typedef struct
{
    struct
    {
        char szBuffer[4096];
    } pClientData[1];
} TERM;
extern TERM Term;
#else // UTM_CLIENT
#include "httpext.h"
#include "tpcc_org.h"
#endif
```

```
#endif

/* FILE: TPCC. H
* Microsoft TPC-C Kit Ver. 3.00.001
* Audited 08/ 23/ 96, By Francois Raab
*
* Copyright Microsoft, 1996
*
* 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               40001
#define _APS_NEXT_CONTROL_VALUE               1000
#define _APS_NEXT_SYMED_VALUE         01
// 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;

#ifdef USE_ODBC
typedef struct _DBPROCESS
{
       HDBC   hdbc;
       HSTMT  hstmt;
       int           pid;
       void   *uPtr;
} DBPROCESS, *PDBPROCESS;
// dblib error message return values
#define INT_EXIT              0
```

```
#define INT_CONTINUE   1
#define INT_CANCEL            2
#endif

// 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
       PDBPROCESS dbproc;     // dblib connection pointer
       int     spid;          // spid assigned from dblib
       int     iSyncId;       // syncronization 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
entries
       int     iNext;                 // next available terminal array
element
       int     iMasterSyncId; // syncronization id
       BOOL    bInit;                 // structure has been initialized
flag
       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 tpcc. dll in the
       // TERMID= key in the HTTP string.
       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
// function prototypes
```

```
BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID
lpReserved);
static void DeliveryDisconnect(void *ptr);
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);
void NumberOfConnectionsCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId);
static void h_printf(EXTENSION_CONTROL_BLOCK *pECB, char *format, ...);
static BOOL GetKeyValue(char *pQueryString, char *pKey, char *pValue,
int iMax);
static void TermInit(void);
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);
BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId);
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,
BOOL bSuccess);
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 void FormatHTMLString(char *szBuff, char *szStr, int iLen);
extern char szErrorLogPath[ 256];
extern EXTENSION_CONTROL_BLOCK *gpECB;


/*  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
        #ifndef TIMESTAMP_STRUCT
            #include <sqltypes.h>
            #include <sql.h>
            #include <sqlext.h>
        #endif
    #else
        #ifndef _INC_SQLFRONT
            #define DBNTWIN32
            #include <sqlfront.h>
            #include <sqldb.h>
        #endif
    #endif

    #ifndef DBINT
        typedef long DBINT;
    #endif

    #define DEFCLPACKSIZE                1024
    #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;
        long                o_id;
        short               o_commit_flag;
    #ifdef USE_ODBC
        TIMESTAMP_STRUCT    o_entry_d;
    #else
        DBDATEREC           o_entry_d;
    #endif
        short               o_all_local;
        double              total_amount;
        long                num_deadlocks;
        char                execution_status[STATUS_LEN];
        OL_NEW_ORDER_DATA Ol[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
        DBDATEREC            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
        DBDATEREC            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
        DBDATEREC           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
        DBDATEREC            o_entry_d;
    #endif
        short                o_carrier_id;
        OL_ORDER_STATUS_DATA
OlOrderStatusData[MAX_OL_ORDER_STATUS_ITEMS];
        short                o_ol_cnt;
        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


#ifndef TPCC_UTIL_H
#define TPCC_UTIL_H
void UtilStrCpy(char *pDest, char *pSrc, int n);
BOOL IsValidTermId(int TermId);
#endif



#ifndef UTM_H_INCLUDED
#define UTM_H_INCLUDED

#ifdef USE_UPIC_CALL
extern int upic_disable(void);
extern int upic_init(void);
extern int upic_call(DWORD dwId, char *service, char *sendbuff, int
sendlen,
                     char *recbuff, int *reclen);
#endif
#define LogFile stderr

#define SERVICE_CHARS 32
typedef union
```

Shared Source Code

```
#include <windows.h>
```

```
{
    NEW_ORDER_DATA      NewOrderData;
    PAYMENT_DATA        PaymentData;
    ORDER_STATUS_DATA   OrderStatusData;
    DELIVERY_DATA       DeliveryData;
    STOCK_LEVEL_DATA    StockLevelData;
    char                ErrorMsg[400];  // ack!!
} TRANS_DATA;

typedef struct
{
    int TermId;
    int SyncId;
    int bDeadlock;
    int bFailed;
    short DeadlockRetry;
    int Error;
    int Return;
    // Note: Trans must be last
    TRANS_DATA Trans;
} UTM_DATA;

typedef struct
{
    char Service[SERVICE_CHARS];
    // Note: Data must be last
    UTM_DATA Data;
} UTM_MSG;

// macros to compute the size of various bits of UTM_MSG. It is
// not enough to just add up the fields because of possible alignment
// issues
#define MSG_HEADER_SIZE(p)((DWORD)(((char *)&(p) ->Data. Trans) -
((char *)(p))))
#define NEW_ORDER_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(NEW_ORDER_DATA))
#define PAYMENT_SIZE(p) ((MSG_HEADER_SIZE((p)) + sizeof(PAYMENT_DATA))
#define ORDER_STATUS_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(ORDER_STATUS_DATA))
#define DELIVERY_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(DELIVERY_DATA))
#define STOCK_LEVEL_SIZE(p) ((MSG_HEADER_SIZE((p)) +
sizeof(STOCK_LEVEL_DATA))
#endif




#include <string.h>
#include <stdio.h>
#include "trans.h"
#include "tpcc.h"
#include "util.h"
#include "error.h"
```

```
char    ErrorMsgBuffer[400] ;

/* 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* pECBpassed in structure pointer
from inetsrv.
* intiErrorid of error message
* intiErrorTypeerror type, ERR_TYPE_SQL, ERR_TYPE_DBLIB, or
ERR_TYPE_WEBDLL
* intiTermIdterminal id from browser
* intiSyncidsync id from browser
* char *szMsgoptional 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 WINAPI 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."},
```

```
        {ERR_D_ID_INVALID,"Invalid District ID Must be 1 to 10."},
        {ERR_MAX_CONNECT_PARAM,"Max client connections exceeded, run
install to increase."},
        {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*\"."},
        {ERR_NEWORDER_DISTRICT_INVALID,"New Order District ID Invalid
range 1 - 10."},
        {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."},
        {ERR_NEWORDER_SUPPW_WITHOUT_ITEMID,"New Order Supp_W field
entered without a corrisponding Item_Id."},
        {ERR_NEWORDER_QTY_WITHOUT_ITEMID,"New Order Qty entered without
a corrisponding 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*\"."},
        {ERR_PAYMENT_HAM_INVALID,"Payment Amount invalid data type must
be numeric."},
        {ERR_PAYMENT_HAM_RANGE,"Payment Amount out of range, 0 -
9999.99."},
        {ERR_ORDERSTATUS_MISSING_DID_KEY,"Order Status missing District
key \"DID*\"."},
        {ERR_ORDERSTATUS_DID_INVALID,"Order Status District invalid,
value must be numeric 1 - 10."},
        {ERR_ORDERSTATUS_DID_RANGE,"Order Status District out of range
must be 1 - 10."},
        {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; */

    szForm = ErrorMsgBuffer ;

                // 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\">");
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"%d\">", iErrorType);
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
                        wsprintf(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\">");
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"%d\">", iErrorType);
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
                        wsprintf(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\">");
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"%d\">", iErrorType);
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
                        wsprintf(szForm+ strlen(szForm), "Error:
DBLIB(%d): %s",iError, szMsg);
                        strcat(szForm, "</FORM><BODY></HTML>");
                        WriteZString(pECB, szForm);
                break;
                case ERR_TYPE_ODBC:
                        strcpy(szForm,"<HTML><HEAD><TITLE> Welcome To
TPC-C</TITLE></HEAD><BODY><FORM ACTION=\"tpcc.dll\"METHOD=\"GET\">");
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"%d\">", iErrorType);
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
                        wsprintf(szForm+ strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
                        wsprintf(szForm+ strlen(szForm), "Error:
ODBC(%d): %s",iError, szMsg);
                        strcat(szForm, "</FORM><BODY></HTML>");
                        WriteZString(pECB, szForm);
                break;
        }
        return;
}

#include <windows.h>
#include <stdio.h>
#include "pipe_routines.h"
#include "trans.h"
#include "utm.h"
const char *SERVER_PIPE_PATH = "\\\\.\\pipe\\tpcc_pipe.%d";
const char *CLIENT_PIPE_PATH = "\\\\.\\pipe\\tpcc_pipe.%d";
```

```

HANDLE DuplicateUtmHandle(HANDLE hSrc, DWORD dwProId)
{
        HANDLE hPro = OpenProcess(PROCESS_DUP_HANDLE, FALSE, dwProId)
;
        HANDLE hDup = NULL ;

        if(hPro)
        {
                if(DuplicateHandle(hPro, hSrc, GetCurrentProcess(),
&hDup, 0, FALSE, DUPLICATE_SAME_ACCESS) == FALSE)
                {
                        Trace( "0x%x: Can not duplicate Handle\n",
GetLastError()) ;
                        CloseHandle(hPro) ;
                }
        }
        else Trace( "0x%x: Can not open Process 0x%x\n",
GetLastError(), dwProId) ;

        return(hDup) ;
}


BOOL OpenClientPipe(SM_PIPE *pPipe, DWORD dwId, UTM_SHARED_MEM
*lpUtmMem)
{
        UTM_HANDLES UtmHandles = lpUtmMem->UtmHandles[dwId] ;

        if((pPipe->evRDav = DuplicateUtmHandle(UtmHandles.evUtmAck,
UtmHandles.dwProId)) &&
              (pPipe->evWDav = DuplicateUtmHandle(UtmHandles.evIisReq,
UtmHandles.dwProId)) &&
              (pPipe->hStop  = DuplicateUtmHandle(UtmHandles.hThread,
UtmHandles.dwProId))   )
        {
                pPipe->dwMaxTransferLen = lpUtmMem->dwMaxTransferLen ;
                pPipe->lpLen = (LPDWORD)((LPBYTE) (&lpUtmMem-
>UtmHandles[lpUtmMem->dwMaxConnections]) + dwId*(lpUtmMem-
>dwMaxTransferLen+sizeof(DWORD))) ;
                pPipe->lpBuffer = ((LPBYTE) pPipe->lpLen) +
sizeof(DWORD) ;
                return(TRUE) ;
        }

        return(FALSE) ;
}


HANDLE CreatePipeEvent(LPSECURITY_ATTRIBUTES lpEventAttributes)
{
        HANDLE hEvent = CreateEvent(lpEventAttributes, FALSE, FALSE,
NULL) ;

        if(!hEvent)
                              Trace( "0x%x: Can not create pipe
event\n", GetLastError) ;

        return(hEvent) ;
}



BOOL OpenServerPipe(SM_PIPE *pPipe, DWORD dwId, LPSECURITY_ATTRIBUTES
lpEventAttributes, UTM_SHARED_MEM *lpUtmMem)
{
        UTM_HANDLES UtmHandles ;


        if((UtmHandles.evIisReq = CreatePipeEvent(lpEventAttributes))
&&
               (UtmHandles.evUtmAck =
CreatePipeEvent(lpEventAttributes))   )
        {
                UtmHandles.hThread =
DuplicateUtmHandle(GetCurrentThread(), GetCurrentProcessId()) ;
                UtmHandles.dwProId = GetCurrentProcessId() ;

                lpUtmMem->UtmHandles[dwId] = UtmHandles ;

                pPipe->evRDav           = UtmHandles.evIisReq ;
                pPipe->evWDav           = UtmHandles.evUtmAck ;
                pPipe->hStop            = DuplicateUtmHandle(lpUtmMem-
>evTerminate, lpUtmMem->dwPIdMasterUtm) ;
                pPipe->dwMaxTransferLen = lpUtmMem->dwMaxTransferLen ;
                pPipe->lpLen            = (LPDWORD)((LPBYTE)
(&lpUtmMem->UtmHandles[lpUtmMem->dwMaxConnections]) + dwId*(lpUtmMem-
>dwMaxTransferLen+sizeof(DWORD))) ;
                pPipe->lpBuffer               = ((LPBYTE) pPipe-
>lpLen) + sizeof(DWORD) ;

                return(TRUE) ;
        }

        return(FALSE) ;
}


BOOL ReadPipe(SM_PIPE *pPipe, void *Buffer, DWORD BufSize, DWORD
*pnRead)
{
        HANDLE Objects[2] = { pPipe->evRDav, pPipe->hStop } ;

        switch(WaitForMultipleObjects(pPipe->hStop ? 2 : 1, Objects,
FALSE, INFINITE))
        {
         case WAIT_OBJECT_0:   // Data is available

                 if(*pPipe->lpLen > BufSize)        // Destination
buffer too small?
                 {
                             Trace( "ReadPipe: buffer too small.Size
was %d, left=%d\n",
                                                              BufSize,
*pPipe->lpLen-BufSize);
                             break ;
                 }

                 *pnRead = *pPipe->lpLen ;
                 CopyMemory(Buffer, pPipe->lpBuffer, *pPipe->lpLen) ;

                 return(TRUE) ;

         case WAIT_OBJECT_0+1:

                 Trace( "ReadPipe: Stop received\n");
                 break ;

         default:

                 Trace( "ReadPipe: Unexpected Wait-State 0x%x\n",
GetLastError());
                 break ;
        }

        *pnRead = 0 ;
        return(FALSE) ;
}



BOOL WritePipe(SM_PIPE *pPipe, void *Buffer, DWORD BytesToWrite, DWORD
*pnWritten)
{
        if(BytesToWrite > pPipe->dwMaxTransferLen)
        {
                 Trace( "WritePipe: buffer too small.Size was %d,
left=%d\n", pPipe->dwMaxTransferLen,


                BytesToWrite-*pPipe->lpLen);
                 *pnWritten = 0 ;
                 return(FALSE) ;
        }

        *pnWritten = *pPipe->lpLen = BytesToWrite ;
        CopyMemory(pPipe->lpBuffer, Buffer, BytesToWrite) ;

        SetEvent(pPipe->evWDav) ;

        return(TRUE) ;
}

#include <windows.h>
#include <string.h>
#include "util.h"
/* 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* pDestdestination string pointer
* char* pSrcsource string pointer
* intnnumber of characters to copy
```

TPCC-DLL Source Code

```
/*      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 <sqldb.h>

#include "delisrv.h"
```

```
*
* RETURNS: None
*
* COMMENTS: Unlike strncpy this function ensures that the result string
is
* always null terminated.
*
*/

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




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;

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
```

```
 *                                   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;
}

/* 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[] = "/-\\|/-\\|";
        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
unsuccessfull
 *                                       ERR_SUCCESS    No error
successfull code
 *
 *
 * COMMENTS:     None
 *
 */

static int Init(void)
{
        int     iError;

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

        fpLog   = NULL;

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

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

        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;
        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
 *                            char    *argv[] array of command line
argument pointers
 *
 * RETURNS:             BOOL    FALSE   parameter read successfull
 *                                              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
 *
 * 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.%2.2d.%3.3d
*\n", versionMS, versionMM, versionLS);
        printf("*                                               *\n");
        printf("*************************************************\n\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
               Reqistry read Successfull, 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)
{
       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 and
 *                     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<iQSlotts; i++)
                {
                        if ( !pDeliveryCache[i].bInUse )
                                break;
                }
                if ( i < iQSlotts )
                {
                        EnterCriticalSection(&DeliveryCriticalSection);
                        pDeliveryCache[i].bInUse = TRUE;
                        LeaveCriticalSection(&DeliveryCriticalSection);
                }
                else
                {
                        EnterCriticalSection(&DeliveryCriticalSection);
                        if ( !(pDeliveryCache =
(LPDELIVERY_PACKET)realloc(pDeliveryCache, sizeof(DELIVERY_PACKET) *
(iQSlotts+512))) )
                        {
                                ErrorMessage(ERR_INSUFFICIENT_MEMORY);

        LeaveCriticalSection(&DeliveryCriticalSection);
                                return;
                        }
                        for(i=iQSlotts; i<iQSlotts+512; i++)
                                pDeliveryCache[i].bInUse = FALSE;
                        i = iQSlotts;
                        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
 *                      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
               operating 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
 *                             int                            msgstate
               message state
 *                             int                            severity
               message severity
 *                             char                   *msgtext
       printable message description
 *
 * RETURNS:            int                            INT_CONTINUE
       continue if error is SQLETIME else INT_CANCEL action
 *                                                            INT_CANCEL
               cancel operation
 *
 * COMMENTS:   This function also sets the dead lock dbproc variable if
necessary.
 *
 */

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 the right database
       dbuse(*dbproc, database);

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

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

       dbsqlexec(*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");

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

       return FALSE;
}


//queue time, end time, elapsed time, w_id, o_carrier_id, o_id1, ...
o_id10
/* FUNCTION: void WriteLog(LPDELIVERY pDelivery)
 *
 * PURPOSE:    This function writes the delivery results to the
delivery log file.
 *
 * ARGUMENTS:  LPDELIVERY     pDelivery      Pointer to delivery
information.
 *
 * RETURNS:           None
 *
 * COMMENTS:   None
 *
 */

static void WriteLog(LPDELIVERY pDelivery)
```

```
{
        int elapsed;

        CalculateElapsedTime(&elapsed, &pDelivery->queue, &pDelivery-
>trans_end);

        EnterCriticalSection(&WriteLogCriticalSection);

        fprintf(fpLog,
"%2.2d/%2.2d/%2.2d,%2.2d:%2.2d:%2.2d:%3.3d,%2.2d:%2.2d:%2.2d:%3.3d,%d,%
d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d,%d\r\n",
                pDelivery->trans_end.wYear - 1900, pDelivery-
>trans_end.wMonth, pDelivery->trans_end.wDay,
                pDelivery->queue.wHour, pDelivery->queue.wMinute,
pDelivery->queue.wSecond, pDelivery->queue.wMilliseconds,
                pDelivery->trans_end.wHour, pDelivery-
>trans_end.wMinute, pDelivery->trans_end.wSecond, pDelivery-
>trans_end.wMilliseconds,
                elapsed,
                pDelivery->w_id, pDelivery->o_carrier_id,
                pDelivery->o_id[0], pDelivery->o_id[1], pDelivery-
>o_id[2], pDelivery->o_id[3],
                pDelivery->o_id[4], pDelivery->o_id[5], pDelivery-
>o_id[6], pDelivery->o_id[7],
                pDelivery->o_id[8], pDelivery->o_id[9] );

        if ( bFlush )
                fflush(fpLog);

        LeaveCriticalSection(&WriteLogCriticalSection);

        return;
}

/* FUNCTION: void CalculateElapsedTime(int *pElapsed, LPSYSTEMTIME
lpBegin, LPSYSTEMTIME lpEnd)
 *
 * PURPOSE:    This function calculates the elapsed time a delivery
transaction took.
 *
 * ARGUMENTS:  int                          *pElapsed      pointer to
int variable to receive calculated elapsed
 *
               time in milliseconds.
 *                            LPSYSTEMTIME   lpBegin        Pointer to
system time structure containing
 *
               transaction beginning time.
 *                            LPSYSTEMTIME   lpEnd          Pointer to
system time structure containing
 *
               transaction ending time.
 * RETURNS:           None
```

```
 *
 * COMMENTS:   None
 *
 */

static void CalculateElapsedTime(int *pElapsed, LPSYSTEMTIME lpBegin,
LPSYSTEMTIME lpEnd)
{
        int                     beginSeconds;
        int                     endSeconds;


        beginSeconds = (lpBegin->wHour * 3600000) + (lpBegin->wMinute *
60000) + (lpBegin->wSecond * 1000) + lpBegin->wMilliseconds;
        endSeconds = (lpEnd->wHour * 3600000) + (lpEnd->wMinute *
60000) + (lpEnd->wSecond * 1000) + lpEnd->wMilliseconds;
        *pElapsed = endSeconds - beginSeconds;

        //check for day boundry, this will function for 24 hour period
however it will not work over 48 hours.
        if ( *pElapsed < 0 )
                *pElapsed = *pElapsed + (24 * 60 * 60 * 1000);

        return;
}

/* FUNCTION: int SQLDelivery(DELIVERY *pDelivery)
 *
 * PURPOSE:    This function processes the delivery transaction.
 *
 * ARGUMENTS:  DELIVERY              *pDelivery             Pointer to
delivery transaction structure
 *
 * RETURNS:            int     ERR_DBGETDATA_FAILED          Delivery
get data operation failed.
 *                                    ERR_SUCCESS
               Delivery successfull, no error
 *
 *
 * COMMENTS:   None
 *
 */

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];
        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;
}

/* FILE: TPCC.C
* Microsoft TPC-C Kit Ver.3.00.000
* Audited 08/23/96By 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>
#include <fcntl.h>
#include "trans.h"   // tpckit transaction header contains definations
of structures specific to TPC-C
#include "httpext.h"// ISAPI DLL information header
#include "tpcc.h"    // this dlls specific structure, value e.t.header.
#include "sqlroutines.h"// the header files for the SQL routines
#include "util.h"
#include "error.h"
#include "pipe_routines.h"

#ifdef USE_ODBC
HENVhenv;
#endif

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 iQSlotts=3000;
int iDelayMs=100;
int iConnectDelay=500;
short iDeadlockRetry=(short) 3;
short iMaxConnections =(short) 25;

#ifdef USE_ODBC
int bConnectionPooling = FALSE;
#endif

// allowable client command strings i.e.CMD= command
char *szCmds[] =
{
    "..NewOrder..", "..Payment..", "..Delivery..", "..Order-Status..",
"..Stock-Level..", "..Exit..",
    "Submit", "Begin", "Process", "Menu", "Clear", "Users", ""
};
// defined command string functions, called via CMD= command http
string from html client.
void (*DoCmd[])(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int iId,
int iSyncId) =
{
    NewOrderForm,
    PaymentForm,
    DeliveryForm,
    OrderStatusForm,
    StockLevelForm,
    Exitcmd,
    SubmitCmd,
    BeginCmd,
    ProcessCmd,
    MenuCmd,
    ClearCmd,
    NumberOfConnectionsCmd
};
// Terminal client id structure and interface defination
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>"
    "<HEAD><TITLE>Welcome To TPC-C</TITLE></HEAD><BODY>"
    "Please Identify your Warehouse and District for this session.<BR>"
    "<FORM ACTION=\"tpcc.dll\"METHOD=\"GET\">"
    "<INPUT TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"0\">"
    "<INPUT TYPE=\"hidden\"NAME=\"FORMID\"VALUE=\"1\">"
    "<INPUT TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"-2\">"
    "<INPUT TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"0\">"
    "Warehouse ID <INPUT NAME=\"w_id\"SIZE=4><BR>"
    "District ID <INPUT NAME=\"d_id\"SIZE=2><BR>"
    "<HR>"
    "<INPUT TYPE=\"submit\"NAME=\"CMD\"VALUE=\"Submit\">"
    "</FORM><BODY>"
    "</HTML>";
static char szTpccLogPath[256]; // path to html log file if logging
turned on in registry.
char szErrorLogPath[256]; // path to error log file.
static CRITICAL_SECTION CriticalSection;
static LPTSTR lpszPipeName=TEXT("\\\\.\\pipe\\DELISRV");
static HANDLE hDeliveryWrite=INVALID_HANDLE_VALUE;
static HANDLE hPipe=INVALID_HANDLE_VALUE;
EXTENSION_CONTROL_BLOCK *gpECB;
static int bTpccExit; // exit delivery disconnect loop as dll exiting.

extern int ThreadCount;

/* 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 baised 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: HANDLEhModulemodule handle
* DWORDul_reason_for_callreason for call
* LPVOIDlpReservedreserved for future use
*
* RETURNS: BOOLFALSEerrors occured in initialization
* TRUEDLL successfully initialized
*
* COMMENTS: None
*
*/
BOOL APIENTRY DllMain(HANDLE hModule, DWORD ul_reason_for_call, LPVOID
lpReserved)
```

```
{
    static SECURITY_ATTRIBUTES sa;
    static PSECURITY_DESCRIPTOR pSD;
    int i = 0;

Trace("maindll reason for call %d\n", (int)ul_reason_for_call);
    switch(ul_reason_for_call)
    {
    case DLL_PROCESS_ATTACH:

#ifdef _DEBUG
             {
          freopen("\\temp\\tpcc.log", "a", stderr);
          setbuf(stderr, NULL);
          Trace("logging started\n");
         }
#endif
             Trace("process attach %d\n", ThreadCount);
        if (ReadRegistrySettings())
        {
            MessageBox(NULL, "Cannot Find TPCC Key in registry (run
install.exe).", "Init", MB_OK | MB_ICONSTOP);
            return FALSE;
        }
        InitializeCriticalSection(&CriticalSection);
        (*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=sizeof(sa);
        sa.lpSecurityDescriptor=pSD;
        sa.bInheritHandle=TRUE;
        // 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(DeliveryDisconnect, 0, NULL) == -1)
        {
            MessageBox(NULL, "Error _beginthread()", "Init", MB_OK |
MB_ICONSTOP);
            return FALSE;
        }
        if (!SQLInit())
            return FALSE;
    break;
    case DLL_THREAD_ATTACH:

             Trace("thread attach %d\n", ThreadCount);

         if (!SQLThreadAttach())
                            return FALSE;
    break;
    case DLL_THREAD_DETACH:

             Trace( "thread %d\n", ThreadCount);

         if (!SQLThreadDetach())
                                    return FALSE;
    break;
    case DLL_PROCESS_DETACH:

             Trace( "process detach %d\n", ThreadCount);

        if (pSD)
            free(pSD);
        bTpccExit = TRUE;
        if (hPipe)
```

```
            DisconnectNamedPipe(hPipe);
        if (hPipe != INVALID_HANDLE_VALUE)
            CloseHandle(hPipe);
        (*Term.Restore)();
        SQLCleanup();
        DeleteCriticalSection(&CriticalSection);

    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* ptrvoid 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(!bTpccExit)
    {
        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* pVerpassed in structure in   which to
place expected version number.
*
* RETURNS: TRUEinet service expected return value.
*
* COMMENTS: None
*
*/
BOOL WINAPI GetExtensionVersion(HSE_VERSION_INFO *pVer)
{
    pVer->dwExtensionVersion = MAKELONG(HSE_VERSION_MINOR,
HSE_VERSION_MAJOR);
    lstrcpyn(pVer->lpszExtensionDesc, "TPC-C Server.",
HSE_MAX_EXT_DLL_NAME_LEN);
    return TRUE;
}

/* FUNCTION: DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK
*pECB)
*
* 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* pECBstructure pointer to passed
in internet
* service information.
*
* RETURNS: DWORDHSE_STATUS_SUCCESSconnection can be dropped if error
* HSE_STATUS_SUCCESS_AND_KEEP_CONNkeep connect valid comment sent
*
* COMMENTS: None
*
*/
DWORD WINAPI HttpExtensionProc(EXTENSION_CONTROL_BLOCK *pECB)
{
    int iCmd, FormId, TermId, iSyncId;
    FILE *fp;

Trace("check Http Thread %d Termid %d\n", ThreadCount,TermId);
```

```
    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: intTermIdclient terminal id
*
* RETURNS: BOOLFALSETerminal ID Invalid
* TRUETerminal ID valid
*
* COMMENTS: None
*
*/
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 relevent information out of the
http command passed in from
* the browser.
*
* ARGUMENTS: EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed
in internet
* service information.
* int* pCmdreturned command id
* int* pFormIdreturned active form client browser is on
* int* pTermIdreturned client terminal id
*
* RETURNS: BOOLFALSEsuccess
* TRUEcommand 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, "TERMID=")))
    {
        *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 (!stricmp(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]);
                }
                break;
            default:
                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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* EXTENSION_CONTROL_BLOCK* pECBstructure 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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*
*/
void DeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int iFormId, int
iTermId, int iSyncId)
{
    WriteZString(pECB, MakeDeliveryForm(iTermId, iSyncId, TRUE, 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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling borwser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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;
}
```

```
/* 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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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);
    return;
}

/* FUNCTION: void ProcessCmd(EXTENSION_CONTROL_BLOCK *pECB, int
iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function process the passed in http command
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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)())
    {
        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: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure 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 NumberOfConnectionsCmd(EXTENSION_CONTROL_BLOCK *pECB,
int iFormId, int iTermId, int iSyncId)
*
* PURPOSE: This function returns to the browser the total number of
active terminal ids
*
* ARGUMENTS: intiFormIdunused
* intiTermIdid of calling browser, i.e.TERMID= from http command line
* intiSyncIdsync id of calling browser
* EXTENSION_CONTROL_BLOCK* pECBstructure pointer to passed in internet
* service information.
* RETURNS: None
*
* COMMENTS: None
*/
void NumberOfConnectionsCmd(EXTENSION_CONTROL_BLOCK *pECB, int iFormId,
int iTermId, int iSyncId)
{
    int i;
    int iTotal;

    // EnterCriticalSection(&CriticalSection);
    iTotal = 0;
    for(i=0; i<Term.iAvailable; i++)
    {
        if (Term.pClientData[i].inUse)
            iTotal++;
    }
    // LeaveCriticalSection(&CriticalSection);
    h_printf(pECB, "Total Active Connections: %d", iTotal);
    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* pECBpassed in structure pointer
from inetsrv.
* char* szStrstring 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.
*/
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\n%s\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;
}

/* 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* pECBpassed in structure pointer
from inetsrv.
* char* formatprintf 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, ...)
{
    char szBuff[512];
    char szTmp[512];
    va_list marker;

    va_start(marker, format);
    vsprintf(szTmp, format, marker);
    va_end(marker);
    wsprintf(szBuff, "<html>%s</html>", szTmp) + 1;
```

```
    WriteZString(pECB, szBuff);
    return;
}

/* FUNCTION: BOOL GetKeyValue(char *pQueryString, char *pKey, char
*pValue, int iMax)
*
* PURPOSE: This function parses a http formatted string for specific
key values.
*
* ARGUMENTS: char* pQueryStringhttp string from client browser
* char* pKeykey value to look for
* char* pValuecharacter array into which to place key’s value
* intiMaxmaximum length of key value array.
*
* RETURNS: BOOLFALSEkey value not found
* TRUEkey valud found
*
*
* COMMENTS: http keys are formatted either KEY=value& or
KEY=value\0.This DLL formats
* TPC-C input fields in such a manner that the keys can be extracted in
the
* above manner.
*/
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 TermInit(void)
*
* PURPOSE: This function initializes the client ternimal structure it
is called when the TPCC.DLL
* is first loaded by the inet service.
*
* ARGUMENTS: none
*
```

```
* RETURNS: None
*
* COMMENTS: None
*
*/
static void TermInit(void)
{
    if (Term.bInit)
    return;
    Term.iNext=0;
    Term.iMasterSyncId=1;
    Term.iAvailable=0;
    Term.pClientData=NULL;
    Term.bInit=TRUE;
    return;
}

/* 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: intTRUE or 1 if sucessfull
* intFALSE or 0 if terminal id cannot be allocated.
*
* 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* pECBpassed in structure pointer
from inetsrv.
* char* pQueryStringhttp query string passed to this DLL.
*
* RETURNS: intassigned terminal id
* -1cannot assign id error occured.
*
*
* 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, iTotalConnections, iTickCount;

    EnterCriticalSection(&CriticalSection);
    for(i=0, iTotalConnections = 0; i<Term.iAvailable; i++)
    {
        if (Term.pClientData[i].inUse)
        iTotalConnections++;
    }
    if (iTotalConnections >= 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].iSyncId = Term.iMasterSyncId++;
    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* pECBpassed in structure pointer
from inetsrv.
* intidTerminal 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* pECBpassed in structure pointer
from inetsrv.
* intiTermIdid of browser client that this connection is for.
* intiSyncIdsync id for this client session
* char* szServersql server name
* char* szUseruser name
* char* szPassworduser password
* char* szDatabasedatabase to use
*
* RETURNS: BOOLFALSEif successfull
* TRUEif 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];
    char server[256];
    char database[256];
    char user[256];
    char password[256];

    sprintf(szApp, "TPCC:%ld", (int) iTermId);
    Term.pClientData[iTermId].dbproc = NULL;
    sprintf(szApp, "TPCC:%ld", (int) iTermId);
    Term.pClientData[iTermId].dbproc = NULL;
    strcpy(server, szServer);
    strcpy(database, szDatabase);
    strcpy(user, szUser);
    strcpy(password, szPassword);
    if (SQLOpenConnection(pECB, iTermId, iSyncId,
&Term.pClientData[iTermId].dbproc,
                            server, database, user, password, 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 *pECBpassed in structure pointer
from inetsrv.
* intiTermIdid of browser client that this connection is for.
* intiSyncIdsync id of client browser
*
* RETURNS: BOOL FALSE if successfull
*                TRUE if an error occurs and connection cannot be
terminated.
*
* COMMENTS: None
*
*/
static BOOL Close(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId)
{
    PECBINFO pEcbInfo;

    if (Term.pClientData[iTermId].dbproc != NULL)
    {
        if ((pEcbInfo = SQLGetECB(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: 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* szDestDestination buffer where formatted string is
to be placed
* char* szPicpicture string which describes how character value is to
be
* formatted.
* char* szSrccharacter 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: intiTermIdclient browser terminal id
* intiSyncIdclient browser sync id
* BOOLbInputTRUE 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=\"\">");
    strcat(szForm, "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"0\">");
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"FORMID\"VALUE=\"%d\">", STOCK_LEVEL_FORM);
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
    strcat(szForm, "<PRE>Stock-Level<BR>");
    wsprintf(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
    {
        wsprintf(szForm+strlen(szForm), "Stock Level Threshold:
%2.2d<BR><BR>",
            Term.pClientData[iTermId].stockLevelData.thresh_hold);
        wsprintf(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)
*
* PURPOSE: This function
*
* ARGUMENTS: intiTermIdclient browser terminal id
* intiSyncIdclient 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\">");
        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
        wsprintf(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: intiTermIdclient browser terminal id
* intiSyncIdclient browser sync id
* BOOLbInputTRUE if form is being constructed for input else FALSE
* BOOLbValidTRUE if NeworderData valid, ELSE FALSE effects output only
*
* 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\">");
    }
    else
    {
    if (bValid)
        strcat(szForm, "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"0\">");
    else
        wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"%d\">", ERR_BAD_ITEM_ID);
    }
    wsprintf(szForm+strlen(szForm),"<INPUT
TYPE=\"hidden\"NAME=\"FORMID\"VALUE=\"%d\">", NEW_ORDER_FORM);
    wsprintf(szForm+strlen(szForm),"<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm),"<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
    strcat(szForm, "<PRE>New Order<BR>");
    if (bInput)
    {
        wsprintf(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=\"Qty00*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP01*\"SIZE=4> <INPUT NAME=\"IID01*\"SIZE=6>
<INPUT NAME=\"Qty01*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP02*\"SIZE=4> <INPUT NAME=\"IID02*\"SIZE=6>
<INPUT NAME=\"Qty02*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP03*\"SIZE=4> <INPUT NAME=\"IID03*\"SIZE=6>
<INPUT NAME=\"Qty03*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP04*\"SIZE=4> <INPUT NAME=\"IID04*\"SIZE=6>
<INPUT NAME=\"Qty04*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP05*\"SIZE=4> <INPUT NAME=\"IID05*\"SIZE=6>
<INPUT NAME=\"Qty05*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP06*\"SIZE=4> <INPUT NAME=\"IID06*\"SIZE=6>
<INPUT NAME=\"Qty06*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP07*\"SIZE=4> <INPUT NAME=\"IID07*\"SIZE=6>
<INPUT NAME=\"Qty07*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP08*\"SIZE=4> <INPUT NAME=\"IID08*\"SIZE=6>
<INPUT NAME=\"Qty08*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP09*\"SIZE=4> <INPUT NAME=\"IID09*\"SIZE=6>
<INPUT NAME=\"Qty09*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP10*\"SIZE=4> <INPUT NAME=\"IID10*\"SIZE=6>
<INPUT NAME=\"Qty10*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP11*\"SIZE=4> <INPUT NAME=\"IID11*\"SIZE=6>
<INPUT NAME=\"Qty11*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP12*\"SIZE=4> <INPUT NAME=\"IID12*\"SIZE=6>
<INPUT NAME=\"Qty12*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP13*\"SIZE=4> <INPUT NAME=\"IID13*\"SIZE=6>
<INPUT NAME=\"Qty13*\"SIZE=1><BR>"
            "<INPUT NAME=\"SP14*\"SIZE=4> <INPUT NAME=\"IID14*\"SIZE=6>
<INPUT NAME=\"Qty14*\"SIZE=1><BR>"
            "Execution Status: Total:<BR><HR>"
            "<INPUT TYPE=\"submit\"NAME=\"CMD\"VALUE=\"Process\">"
            "<INPUT TYPE=\"submit\"NAME=\"CMD\"VALUE=\"Menu\">"
            "</FORM>"
            "</HTML>");
    }
    else
    {
        if (bValid)
        {
            wsprintf(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
        {
        wsprintf(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);
        wsprintf(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.o_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,
                    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: intiTermIdclient browser terminal id
* intiSyncIdclient browser sync id
* BOOLbInputTRUE 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=\"0\">");
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"FORMID\"VALUE=\"%d\">", PAYMENT_FORM);
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
```

```
    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=\"CWI*\"SIZE=4>"
            "Cust-District: <INPUT NAME=\"CDI*\"SIZE=1><BR>"
            "Name: <INPUT NAME=\"CLT*\"SIZE=16> Since:<BR>"
            "Credit:<BR>"
            "Disc:<BR>"
            "Phone:<BR><BR>"
            "Amount Paid: $<INPUT NAME=\"HAM*\"SIZE=7> New Cust
Balance:<BR>"
            "Credit Limit:<BR><BR>Cust-Data:<BR><BR><BR><BR></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 %%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>",
        szTmpStr1, szTmpStr2, szC_Zip, szC_Phone);
    sprintf(szForm+strlen(szForm), "Amount Paid:$%7.2f New Cust
Balance: $%4.2f<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 <= l)
                UtilStrCpy(szTmp, ptr, 50);
            else
                szTmp[0] = 0;
            if (!i)
            {
                FormatHTMLString(szTmpStr1, szTmp, 50);
                wsprintf(szForm+strlen(szForm), "Cust-Data: %s<BR>",
szTmpStr1);
            }
            else
            {
                FormatHTMLString(szTmpStr1, szTmp, 50);
                wsprintf(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: intiTermIdclient browser terminal id
* intiSyncIdclient browser sync id
* BOOLbInputTRUE 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\">");
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"FORMID\"VALUE=\"%d\">", ORDER_STATUS_FORM);
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
    strcat(szForm,"<PRE>Order-Status<BR>");
    wsprintf(szForm+strlen(szForm), "Warehouse: %4.4d ",
        Term.pClientData[iTermId].orderStatusData.w_id);
    if (bInput)
    {
        strcat(szForm,"District: <INPUT NAME=\"DID*\"SIZE=1><BR>"
            "Customer: <INPUT NAME=\"CID*\"SIZE=4> Name: <INPUT
NAME=\"CLT*\"SIZE=23><BR>"
            "Cust-Balance:<BR><BR>"
            "Order-Number: Entry-Date: Carrier-Number:<BR>"
            "Supply-W Item-Id Qty Amount Delivery-Date<BR></PRE>"
            "<HR><INPUT
TYPE=\"submit\"NAME=\"CMD\"VALUE=\"Process\"><INPUT
TYPE=\"submit\"NAME=\"CMD\"VALUE=\"Menu\">"
            "</BODY></FORM></HTML>");
    }
    else
    {
        wsprintf(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);
        wsprintf(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);
        wsprintf(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.OlOrderStatusData[i].ol_suppl
y_w_id,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_i_id,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_quant
ity,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_amoun
t,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliv
ery_d.day,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliv
ery_d.month,

Term.pClientData[iTermId].orderStatusData.OlOrderStatusData[i].ol_deliv
ery_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, BOOL bSuccess)
*
* PURPOSE: This function
*
* ARGUMENTS: intiTermIdclient browser terminal id
* intiSyncIdclient browser sync id
* BOOLbInputTRUE if form is being constructed for input else FALSE
* BOOLbSuccess TRUE if Delivery succeeded 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,
BOOL bSuccess)
{
    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\">");
    }
    else
    {
        if (!bSuccess)
            sprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"%d\">",
ERR_TYPE_DELIVERY_POST);
        else
            strcat(szForm, "<INPUT
TYPE=\"hidden\"NAME=\"STATUSID\"VALUE=\"0\">");
    }
```

```
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"FORMID\"VALUE=\"%d\">", DELIVERY_FORM);
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"TERMID\"VALUE=\"%d\">", iTermId);
    wsprintf(szForm+strlen(szForm), "<INPUT
TYPE=\"hidden\"NAME=\"SYNCID\"VALUE=\"%d\">", iSyncId);
    strcat(szForm,"<PRE>Delivery<BR>");
    wsprintf(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
    {
        wsprintf(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
    {
        wsprintf(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 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* pECBpassed in structure pointer
from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncId 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;
    }
    iRc = SQLNewOrder(pECB, iTermId, iSyncId,
        Term.pClientData[iTermId].dbproc,
        &Term.pClientData[iTermId].newOrderData, iDeadlockRetry);

    #ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    if (bConnectionPooling && iRc != -3)
    SQLDisconnect(Term.pClientData[iTermId].dbproc->hdbc);
    #endif
    #endif

    if ((pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)) &&
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* pECBpassed in structure pointer
from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncId 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);

    #ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    if (bConnectionPooling && iRc != -3)
    SQLDisconnect(Term.pClientData[iTermId].dbproc->hdbc);
    #endif
    #endif

    if ((pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)) &&
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* pECBpassed in structure pointer
from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncId client browser sync id
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void ProcessOrderStatusForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
    int iRc;
    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);

    #ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    if (bConnectionPooling && iRc != -3)
    SQLDisconnect(Term.pClientData[iTermId].dbproc->hdbc);
    #endif
    #endif

    if ((pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)) &&
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* pECBpassed in structure pointer
from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncIdclinet browser sync id
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void ProcessDeliveryForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
    char szTmp[26];
    BOOL bSuccess;

    memset(&Term.pClientData[iTermId].deliveryData, 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");
        bSuccess = FALSE;
    }
    else
    {
        strcpy(Term.pClientData[iTermId].deliveryData.execution_status,
"Delivery has been queued.");
        bSuccess = TRUE;
    }
    WriteZString(pECB, MakeDeliveryForm(iTermId, iSyncId, FALSE,
bSuccess));
    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
* 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* pECBpassed in structure pointer
from inetsrv.
* intiTermIdclient browser terminal id
* intiSyncIdclient browser sync id
*
* RETURNS: None
*
* COMMENTS: None
*
*/
static void ProcessStockLevelForm(EXTENSION_CONTROL_BLOCK *pECB, int
iTermId, int iSyncId)
{
    char szTmp[26];
    int iRc;
    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;
    }
    iRc = SQLStockLevel(pECB, iTermId, iSyncId,
        Term.pClientData[iTermId].dbproc,
        &Term.pClientData[iTermId].stockLevelData, iDeadlockRetry);

    #ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    if (bConnectionPooling && iRc != -3)
    SQLDisconnect(Term.pClientData[iTermId].dbproc->hdbc);
    #endif
    #endif

    if ((pEcbInfo = SQLGetECB(Term.pClientData[iTermId].dbproc)) &&
pEcbInfo->bFailed)
        return;
    if (iRc)
```

```
        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: LPSTRlpszQueryStringclient browser http command string
* NEW_ORDER_DATA* pNewOrderDatapointer to new order data structure
*
* RETURNS: interror code indicating reason for failure
* ERR_SUCCESSnew 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;
    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: LPSTRlpszQueryStringclient browser http command string
* PAYMENT_DATA* pPaymentDatapointer to payment data structure
*
* RETURNS: interror code indicating reason for failure
* ERR_SUCCESSall input data successfully parsed
*
* COMMENTS: None
*
*/
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.
*
* ARGUMENTS: LPSTRlpszQueryStringclient browser http command string
* ORDER_STATUS_DATA* pOrderStatusDatapointer to order status data
structure
*
* RETURNS: interror code indicating reason for failure
* ERR_SUCCESSsuccessfully parsed all required input data
*
* COMMENTS: None
*
*/
static int GetOrderStatusData(LPSTR lpszQueryString, ORDER_STATUS_DATA
*pOrderStatusData)
{
    char szTmp[26];

    if (!GetKeyValue(lpszQueryString, "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(lpszQueryString, "CID*", szTmp, sizeof(szTmp)))
        return ERR_ORDERSTATUS_MISSING_CID_KEY;
    if (szTmp[0] == 0)
    {
        pOrderStatusData->c_id = 0;
        if (!GetKeyValue(lpszQueryString, "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(lpszQueryString, "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;
    iQSlotts=3000;
    iDelayMs=100;
    iDeadlockRetry=(short)3;
    strcpy(szTpccLogPath, "tpcclog.");

    #ifdef USE_ODBC
    bConnectionPooling = FALSE;
    #endif

    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);
    if (RegQueryValueEx(hKey, "LOG", 0, &type, szTmp,   &size) ==
ERROR_SUCCESS)
    {
        if (!stricmp(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, "QueueSlotts", 0, &type, szTmp, &size) ==
ERROR_SUCCESS)
        iQSlotts = atoi(szTmp);
    if (!iQSlotts)
```

```
        iQSlotts = 3000;
    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;

    #ifdef USE_ODBC
    #if (ODBCVER >= 0x0300)
    size = sizeof(szTmp);
    if (RegQueryValueEx(hKey, "ConnectionPooling", 0, &type, szTmp,
&size) == ERROR_SUCCESS)
    if (!stricmp(szTmp, "ON"))
    bConnectionPooling = TRUE;
    iConnectDelay = 500;
    size = sizeof(szTmp);
    if (RegQueryValueEx(hKey, "ConnectionPoolRetryTime", 0, &type,
szTmp, &size) == ERROR_SUCCESS)
    iConnectDelay = atoi(szTmp);
    if (!iConnectDelay)
    iConnectDelay = 500;
    #endif
    #endif

    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: shortw_idwarehouse id
* shorto_carrier_idcarrier id
*
* RETURNS: BOOLFALSEdelivery information posted successfully
* TRUEerror 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* ptrpointer to string to check.
*
* RETURNS: BOOLFALSEif string is not all numeric
* TRUEif 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);
}

/* 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* szBuffReturned string information
* char* szStrinput string to be formatted.
* intiLenLength 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.
*
*/
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;
}

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

#ifdef USE_ODBC
#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>
HENV henv;
#else

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

#include "trans.h"
#include "httpext.h"
#include "tpcc.h"
#include "utm.h"
#include "sqlroutines.h"
#include "pipe_routines.h"
#include "util.h"
const int ARGSIZE= 1024;
const int PIPE_BUF_SIZE= 4096;

static CRITICAL_SECTION CriticalSection;
void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr);

typedef struct
{
        DWORD   dwId;
        SM_PIPE   Pipe;
} THREAD_DATA;

UTM_SHARED_MEM *lpUtmMem ;
HANDLE           hUtmMem ;
```

```
DWORD dwRingBufferRd ;
DWORD dwRingBufferWrt ;
DWORD *pFreePipeBuffers ;

DWORD TlsIndex;
DWORD ThreadCount= 0;


DWORD GetPipeIndex()
{
        DWORD dwIndex = pFreePipeBuffers[dwRingBufferRd++] ;

        if(dwRingBufferRd == lpUtmMem->dwMaxConnections)
         dwRingBufferRd = 0 ;

        ThreadCount++ ;

        return(dwIndex) ;
}

void PushPipeIndex(DWORD dwId)
{
        EnterCriticalSection(&CriticalSection) ;

        pFreePipeBuffers[dwRingBufferWrt++] = dwId ;

        if(dwRingBufferWrt == lpUtmMem->dwMaxConnections)
         dwRingBufferWrt = 0 ;

        ThreadCount-- ;

        LeaveCriticalSection(&CriticalSection) ;
}


void CloseClientPipe(THREAD_DATA *pData)
{
        if(pData->Pipe.evRDav)
                                                CloseHandle(pData-
>Pipe.evRDav) ;

        if(pData->Pipe.evWDav)
                                                CloseHandle(pData-
>Pipe.evWDav) ;

        if(pData->Pipe.hStop)
                                               CloseHandle(pData-
>Pipe.hStop) ;

        PushPipeIndex(pData->dwId) ;
}
```

```
BOOL SQLThreadAttach(void)
{
        THREAD_DATA *pData;

        Trace( "SQLThread attach starts\n");

        pData = (THREAD_DATA *) malloc(sizeof(THREAD_DATA));
        if (!pData)
                              return FALSE;

        memset(pData, 0, sizeof(*pData));

        EnterCriticalSection(&CriticalSection);

        if(ThreadCount >= lpUtmMem->dwMaxConnections)
        {
                Trace( "SQLThreadattach failed because all SM-Pipes
are in use\n");
                free(pData);

                LeaveCriticalSection(&CriticalSection);
                return FALSE;
        }

        pData->dwId = GetPipeIndex() ;

        LeaveCriticalSection(&CriticalSection);

        if(!OpenClientPipe(&pData->Pipe, pData->dwId, lpUtmMem))
        {
                CloseClientPipe(pData) ;

                free(pData) ;

                TlsSetValue(TlsIndex, 0) ;

                Trace( "SQLThreadattach failed for thread %d\n",
pData->dwId);

                return(FALSE) ;
        }

        TlsSetValue(TlsIndex, pData);

        return(TRUE) ;
}


BOOL SQLThreadDetach(void)
{
        THREAD_DATA *pData = TlsGetValue(TlsIndex);

        if (pData)
        {
                CloseClientPipe(pData);
                free(pData);
        }

        return TRUE;
}


BOOL SQLInit(void)
{
// Perform one time initialization.According to the comments in tpcc.c,
this will
// be called once when the DLL is loaded.We assume that is true, and
also that
// the caller has protected the call with a critical section.
        InitializeCriticalSection(&CriticalSection);

        TlsIndex = TlsAlloc();
        if (TlsIndex == 0xffffffff)
        {
                MessageBox(NULL, "TlsAlloc failed", "Init", MB_OK |
MB_ICONSTOP);
                return FALSE;
        }

        {
                HANDLE evUtmMemInit = OpenEvent(SYNCHRONIZE, FALSE,
UTM_MEM_EVENT) ;

                if(!evUtmMemInit)
                {
                        Trace("0x%x: Can not open synchronize
event\n", GetLastError()) ;

                        return(FALSE) ;
                }

                switch(WaitForSingleObject(evUtmMemInit, 5*60*1000))
                {
                        case WAIT_OBJECT_0:

                                      break ;

                        case WAIT_TIMEOUT:

                                      CloseHandle(evUtmMemInit) ;

                                      Trace("utm_client is not
ready\n") ;
                                      return(FALSE) ;

                        default:
```

```
                                      CloseHandle(evUtmMemInit) ;

                                      Trace("0x%x: Can not
synchronize\n") ;
                                      return(FALSE) ;
                 }

                CloseHandle(evUtmMemInit) ;
        }

        hUtmMem = OpenFileMapping(FILE_MAP_ALL_ACCESS, FALSE,
UTM_MEM_SPACE) ;

        if(hUtmMem == NULL)
                                             return(FALSE) ;

        lpUtmMem = MapViewOfFile(hUtmMem, FILE_MAP_ALL_ACCESS, 0, 0,
0) ;

        if(lpUtmMem)
        {
                DWORD dwI ;

                pFreePipeBuffers = malloc(lpUtmMem->dwMaxConnections *
sizeof(DWORD)) ;

                if(!pFreePipeBuffers)
                                                        return(FALSE) ;

                for(dwI=0; dwI<lpUtmMem->dwMaxConnections; dwI++)

                                        pFreePipeBuffers[dwI] = dwI ;

                dwRingBufferRd = dwRingBufferWrt = 0 ;
        }
        else return(FALSE) ;

        Trace( "TlsIndex = %d\n", TlsIndex);

        return(TRUE) ;
}


void SQLCleanup(void)
{
        if(lpUtmMem)
                               UnmapViewOfFile(lpUtmMem) ;

        if(hUtmMem)
        {
                CloseHandle(hUtmMem) ;
                 hUtmMem = NULL ;
                 lpUtmMem = NULL ;
        }

        TlsFree(TlsIndex);
        TlsIndex = 0xffffffff;
        DeleteCriticalSection(&CriticalSection);
}


BOOL SQLOpenConnection(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId,
                       DBPROCESS **dbproc, char *server, char
*database,
                       char *user, char *password, char *app, int
*spid)
{
    PECBINFO pEcbInfo;
    // set pECB data into dbproc
    pEcbInfo = (PECBINFO) malloc(sizeof(ECBINFO));
    pEcbInfo->bDeadlock = FALSE;
    pEcbInfo->pECB= pECB;
    pEcbInfo->iTermId= iTermId;
    pEcbInfo->iSyncId= iSyncId;
    *dbproc = (DBPROCESS *) pEcbInfo;
    return FALSE;
}


BOOL SQLCloseConnection(EXTENSION_CONTROL_BLOCK *pECB, DBPROCESS
*dbproc)
{
    return FALSE;
}


BOOL UTMTransaction(char *Service, EXTENSION_CONTROL_BLOCK *pECB,
                    int TermId, int SyncId, DBPROCESS *dbproc,
                    short DeadlockRetry, void *Data, long BufSize)
{
    THREAD_DATA *pData;
    UTM_MSG msg;
    DWORD nBytes;

    PECBINFO pECBInfo = (PECBINFO) dbproc;
    // forgive them them, for they know not what they do...
    // we are pessimistic here
    pECBInfo->bFailed = TRUE;
    pData = TlsGetValue(TlsIndex);
    if (pData == NULL)
    {
        if (!SQLThreadAttach())
        {
            Trace( "UTMTransaction: unable to attach\n");
            return FALSE;
        }
        pData = TlsGetValue(TlsIndex);
    }
    // fill the struct to ship to tm
    strcpy(msg.Service, Service);
    msg.Data.TermId = TermId;
    msg.Data.SyncId = SyncId;
    msg.Data.DeadlockRetry = DeadlockRetry;
    msg.Data.Error = FALSE;
    memcpy(&msg.Data.Trans, Data, BufSize);
    if (!WritePipe(&pData->Pipe, &msg, MSG_HEADER_SIZE(&msg)+ BufSize,
&nBytes))
    {
        Trace( "UTMtransaction: WritePipe Failed\n");
        return FALSE;
    }
    if (nBytes != MSG_HEADER_SIZE(&msg)+ BufSize)
    {
        Trace( "UTMtransaction: short write, size=%d, written=%d\n",
                         MSG_HEADER_SIZE(&msg)+ BufSize, nBytes);
        return FALSE;
    }
    if (!ReadPipe(&pData->Pipe, &msg, sizeof(msg), &nBytes))
    {
        Trace( "UTMtransaction: ReadPipe Failed\n");
        return FALSE;
    }
    if (msg.Data.Error)
    {
#ifdef _DEBUG
Trace( "msg.Error set, ErrorMsg=%s\n", msg.Data.Trans.ErrorMsg);
#endif
        WriteZString(pECB, msg.Data.Trans.ErrorMsg);
    }
    // patch things up so the upper levels don't know this went
    // through tm
    pECBInfo->iTermId = TermId;
    pECBInfo->iSyncId = SyncId;
    pECBInfo->bDeadlock = msg.Data.bDeadlock;
    pECBInfo->bFailed = msg.Data.bFailed;
#ifdef _DEBUG
Trace( "bFailed=%d\n", pECBInfo->bFailed);
#endif
    memcpy(Data, &msg.Data.Trans, BufSize);
    return msg.Data.Return;
}


BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId,
                   DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel,
short deadlock_retry)
```

```
{
    long ReceiveLen = sizeof(STOCK_LEVEL_DATA);

    return UTMTransaction("STOCK_LEVEL", pECB, iTermId,
                            iSyncId, dbproc, deadlock_retry,
pStockLevel,
                            sizeof(*pStockLevel));
}


int SQLNewOrder(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId,
                DBPROCESS *dbproc, NEW_ORDER_DATA *pNewOrder, short
deadlock_retry)
{

    return UTMTransaction("NEW_ORDER", pECB, iTermId,
                            iSyncId, dbproc, deadlock_retry, pNewOrder,
                            sizeof(*pNewOrder));
}


int SQLPayment(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int iSyncId,
                DBPROCESS *dbproc, PAYMENT_DATA *pPayment, short
deadlock_retry)
{
    return UTMTransaction("PAYMENT", pECB, iTermId,
                            iSyncId, dbproc, deadlock_retry, pPayment,
                            sizeof(*pPayment));
}


int SQLOrderStatus(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId,
                DBPROCESS *dbproc, ORDER_STATUS_DATA *pOrderStatus,
short deadlock_retry)
{
    return UTMTransaction("ORDER_STATUS", pECB, iTermId,
                            iSyncId, dbproc, deadlock_retry,
pOrderStatus,
                            sizeof(*pOrderStatus));
}


PECBINFO SQLGetECB(PDBPROCESS p)
{
    return (PECBINFO) p;
}

LIBRARY TPCC.DLL

EXPORTS
```

```
        GetExtensionVersion    @1
        HttpExtensionProc      @2

//Microsoft Developer Studio generated resource script.
//
#include "resource.h"

#define APSTUDIO_READONLY_SYMBOLS
///////////////////////////////////////////////////////////////////////
//////
//
// Generated from the TEXTINCLUDE 2 resource.
//
#include "afxres.h"

///////////////////////////////////////////////////////////////////////
//////
#undef APSTUDIO_READONLY_SYMBOLS

///////////////////////////////////////////////////////////////////////
//////
// English (U.S.) resources

#if !defined(AFX_RESOURCE_DLL) || defined(AFX_TARG_ENU)
#ifdef _WIN32
LANGUAGE LANG_ENGLISH, SUBLANG_ENGLISH_US
#pragma code_page(1252)
#endif //_WIN32

#ifndef _MAC
///////////////////////////////////////////////////////////////////////
//////
//
// Version
//

VS_VERSION_INFO VERSIONINFO
 FILEVERSION 0,3,0,2
 PRODUCTVERSION 0,3,0,2
 FILEFLAGSMASK 0x3fL
#ifdef _DEBUG
 FILEFLAGS 0x1L
#else
 FILEFLAGS 0x0L
#endif
 FILEOS 0x40004L
 FILETYPE 0x2L
 FILESUBTYPE 0x0L
BEGIN
    BLOCK "StringFileInfo"
    BEGIN
        BLOCK "040904b0"
        BEGIN
```

```
            VALUE "Comments", "TPC-C HTML DLL Server\0"
            VALUE "CompanyName", "Microsoft\0"
            VALUE "FileDescription", "tpcc\0"
            VALUE "FileVersion", "0, 3, 0, 2\0"
            VALUE "InternalName", "tpcc\0"
            VALUE "LegalCopyright", "Copyright © 1996\0"
            VALUE "OriginalFilename", "tpcc.dll\0"
            VALUE "ProductName", "Microsoft tpcc\0"
            VALUE "ProductVersion", "0, 3, 0, 2\0"
        END
    END
    BLOCK "VarFileInfo"
    BEGIN
        VALUE "Translation", 0x409, 1200
    END
END

#endif    // !_MAC


#ifdef APSTUDIO_INVOKED
///////////////////////////////////////////////////////////////////////
//////
//
// TEXTINCLUDE
//

1 TEXTINCLUDE DISCARDABLE
BEGIN
    "resource.h\0"
END

2 TEXTINCLUDE DISCARDABLE
BEGIN
    "#include ""afxres.h""\r\n"
    "\0"
END

3 TEXTINCLUDE DISCARDABLE
BEGIN
    "\r\n"
    "\0"
END

#endif    // APSTUDIO_INVOKED

#endif    // English (U.S.) resources
///////////////////////////////////////////////////////////////////////
//////



#ifndef APSTUDIO_INVOKED
```

```
///////////////////////////////////////////////////////////////////////
//////
//
// Generated from the TEXTINCLUDE 3 resource.
//


///////////////////////////////////////////////////////////////////////
//////
#endif    // not APSTUDIO_INVOKED


# Microsoft Developer Studio Generated NMAKE File, Format Version 4.10
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Dynamic-Link Library" 0x0102

!IF "$(CFG)" == ""
CFG=tpcc - Win32 Debug
!MESSAGE No configuration specified.  Defaulting to tpcc - Win32 Debug.
!ENDIF

!IF "$(CFG)" != "tpcc - Win32 Release" && "$(CFG)" != "tpcc - Win32
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 /f "tpcc.mak" CFG="tpcc - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "tpcc - Win32 Release" (based on "Win32 (x86) Dynamic-Link
Library")
!MESSAGE "tpcc - Win32 Debug" (based on "Win32 (x86) Dynamic-Link
Library")
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF

!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
#######################################################################
#########
# Begin Project
# PROP Target_Last_Scanned "tpcc - Win32 Debug"
MTL=mktyplib.exe
CPP=cl.exe
RSC=rc.exe
```

```
!IF  "$(CFG)" == "tpcc - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
OUTDIR=.\Release
INTDIR=.\Release

ALL : "$(OUTDIR)\tpcc.dll"

CLEAN :
        -@erase "$(INTDIR)\error.obj"
        -@erase "$(INTDIR)\pipe_routines.obj"
        -@erase "$(INTDIR)\tpcc.obj"
        -@erase "$(INTDIR)\TPCC.res"
        -@erase "$(INTDIR)\util.obj"
        -@erase "$(INTDIR)\utm_sql.obj"
        -@erase "$(OUTDIR)\tpcc.dll"
        -@erase "$(OUTDIR)\tpcc.exp"
        -@erase "$(OUTDIR)\tpcc.lib"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D
"_WINDOWS" /YX /c
# ADD CPP /nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS"
/YX /c
CPP_PROJ=/nologo /MT /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_WINDOWS"\
 /Fp"$(INTDIR)/tpcc.pch" /YX /Fo"$(INTDIR)/" /c
CPP_OBJS=.\Release/
CPP_SBRS=.\.
# ADD BASE MTL /nologo /D "NDEBUG" /win32
# ADD MTL /nologo /D "NDEBUG" /win32
MTL_PROJ=/nologo /D "NDEBUG" /win32
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
RSC_PROJ=/l 0x409 /fo"$(INTDIR)/TPCC.res" /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/tpcc.bsc"
BSC32_SBRS= \

LINK32=link.exe
```

```
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /machine:I386
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib /nologo /subsystem:windows /dll /incremental:no\
 /pdb:"$(OUTDIR)/tpcc.pdb" /machine:I386 /def:".\TPCC.DEF"\
 /out:"$(OUTDIR)/tpcc.dll" /implib:"$(OUTDIR)/tpcc.lib"
DEF_FILE= \
        ".\TPCC.DEF"
LINK32_OBJS= \
        "$(INTDIR)\error.obj" \
        "$(INTDIR)\pipe_routines.obj" \
        "$(INTDIR)\tpcc.obj" \
        "$(INTDIR)\TPCC.res" \
        "$(INTDIR)\util.obj" \
        "$(INTDIR)\utm_sql.obj"

"$(OUTDIR)\tpcc.dll" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
OUTDIR=.\Debug
INTDIR=.\Debug

ALL : "$(OUTDIR)\tpcc.dll"

CLEAN :
        -@erase "$(INTDIR)\error.obj"
        -@erase "$(INTDIR)\pipe_routines.obj"
        -@erase "$(INTDIR)\tpcc.obj"
        -@erase "$(INTDIR)\TPCC.res"
        -@erase "$(INTDIR)\util.obj"
        -@erase "$(INTDIR)\utm_sql.obj"
        -@erase "$(INTDIR)\vc40.idb"
        -@erase "$(INTDIR)\vc40.pdb"
        -@erase "$(OUTDIR)\tpcc.dll"
        -@erase "$(OUTDIR)\tpcc.exp"
        -@erase "$(OUTDIR)\tpcc.ilk"
        -@erase "$(OUTDIR)\tpcc.lib"
        -@erase "$(OUTDIR)\tpcc.pdb"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG"
/D "_WINDOWS" /YX /c
# ADD CPP /nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_WINDOWS" /YX /c
CPP_PROJ=/nologo /MTd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_WINDOWS"\
 /Fp"$(INTDIR)/tpcc.pch" /YX /Fo"$(INTDIR)/" /Fd"$(INTDIR)/" /c
CPP_OBJS=.\Debug/
CPP_SBRS=.\.
# ADD BASE MTL /nologo /D "_DEBUG" /win32
# ADD MTL /nologo /D "_DEBUG" /win32
MTL_PROJ=/nologo /D "_DEBUG" /win32
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
RSC_PROJ=/l 0x409 /fo"$(INTDIR)/TPCC.res" /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/tpcc.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /debug
/machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:windows /dll /debug
/machine:I386
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib /nologo /subsystem:windows /dll /incremental:yes\
 /pdb:"$(OUTDIR)/tpcc.pdb" /debug /machine:I386 /def:".\TPCC.DEF"\
 /out:"$(OUTDIR)/tpcc.dll" /implib:"$(OUTDIR)/tpcc.lib"
DEF_FILE= \
        ".\TPCC.DEF"
LINK32_OBJS= \
        "$(INTDIR)\error.obj" \
        "$(INTDIR)\pipe_routines.obj" \
        "$(INTDIR)\tpcc.obj" \
        "$(INTDIR)\TPCC.res" \
        "$(INTDIR)\util.obj" \
```

```
        "$(INTDIR)\utm_sql.obj"

"$(OUTDIR)\tpcc.dll" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ENDIF

.c{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.c{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

#######################################################################
#########
# Begin Target

# Name "tpcc - Win32 Release"
# Name "tpcc - Win32 Debug"

!IF  "$(CFG)" == "tpcc - Win32 Release"

!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"

!ENDIF

#######################################################################
#########
# Begin Source File

SOURCE=.\tpcc.c
DEP_CPP_TPCC_=\
        {$(INCLUDE)}"\pipe_routines.h"\
        {$(INCLUDE)}"\sqldb.h"\
        {$(INCLUDE)}"\sqlfront.h"\
        {$(INCLUDE)}"\sqlroutines.h"\
        {$(INCLUDE)}"\tpcc.h"\
        {$(INCLUDE)}"\tpcc_org.h"\
        {$(INCLUDE)}"\trans.h"\
        {$(INCLUDE)}"\util.h"\


"$(INTDIR)\tpcc.obj" : $(SOURCE) $(DEP_CPP_TPCC_) "$(INTDIR)"


# End Source File
#######################################################################
#########
# Begin Source File

SOURCE="\openUTM-SRC\AUDIT\shared\error.c"
DEP_CPP_ERROR=\
        {$(INCLUDE)}"\sqldb.h"\
        {$(INCLUDE)}"\sqlfront.h"\
        {$(INCLUDE)}"\tpcc.h"\
        {$(INCLUDE)}"\tpcc_org.h"\
        {$(INCLUDE)}"\trans.h"\
        {$(INCLUDE)}"\util.h"\


"$(INTDIR)\error.obj" : $(SOURCE) $(DEP_CPP_ERROR) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)


# End Source File
#######################################################################
#########
# Begin Source File

SOURCE="\openUTM-SRC\AUDIT\shared\pipe_routines.c"
DEP_CPP_PIPE_=\
        {$(INCLUDE)}"\pipe_routines.h"\
        {$(INCLUDE)}"\sqldb.h"\
        {$(INCLUDE)}"\sqlfront.h"\
        {$(INCLUDE)}"\trans.h"\
        {$(INCLUDE)}"\utm.h"\


"$(INTDIR)\pipe_routines.obj" : $(SOURCE) $(DEP_CPP_PIPE_) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)


# End Source File
#######################################################################
#########
# Begin Source File

SOURCE="\openUTM-SRC\AUDIT\shared\util.c"
DEP_CPP_UTIL_=\
        {$(INCLUDE)}"\util.h"\
```

```
"$(INTDIR)\util.obj" : $(SOURCE) $(DEP_CPP_UTIL_) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)


# End Source File
#######################################################################
#########
# Begin Source File

SOURCE=.\utm_sql.c
DEP_CPP_UTM_S=\
        {$(INCLUDE)}"\pipe_routines.h"\
        {$(INCLUDE)}"\sqldb.h"\
        {$(INCLUDE)}"\sqlfront.h"\
        {$(INCLUDE)}"\sqlroutines.h"\
        {$(INCLUDE)}"\tpcc.h"\
        {$(INCLUDE)}"\tpcc_org.h"\
        {$(INCLUDE)}"\trans.h"\
        {$(INCLUDE)}"\util.h"\
        {$(INCLUDE)}"\utm.h"\


"$(INTDIR)\utm_sql.obj" : $(SOURCE) $(DEP_CPP_UTM_S) "$(INTDIR)"


# End Source File
#######################################################################
#########
# Begin Source File

SOURCE=.\TPCC.DEF

!IF  "$(CFG)" == "tpcc - Win32 Release"

!ELSEIF  "$(CFG)" == "tpcc - Win32 Debug"

!ENDIF

# End Source File
#######################################################################
#########
# Begin Source File

SOURCE=.\TPCC.RC

"$(INTDIR)\TPCC.res" : $(SOURCE) "$(INTDIR)"
   $(RSC) $(RSC_PROJ) $(SOURCE)


# End Source File
# End Target
# End Project
```

```
########################################################################
#########
```

Client Application Source Code

```
/* link xtclt32.lib or upicw32.lib */
/*
*
* UTM Client   utm_client.c
*
* main transaction client process to start all utm threads for
* listen on tpcc-pipes and send the request to transaction server
*
* usage: utm_client <process number> [<number of threads>]
*        if the second value is not set, the default max threads per
*        process is used
*
*/
/* Johann Gebendorfer, MW TP QA, 10.9.97
   Aenderungen fuer die Umsetzung von xatmi-Aufrufen auf upic-Aufrufen
   Wird USE_UPIC_CALL definiert, dann ergeben sich folgende Aenderungen
   (wirksam durch #ifdef USE_UPIC_CALL )
    - UTMBuffer wird statt als Pointer als Array definiert
      !!!! unklar: wie gross soll der Array definiert werden !!!!
       tpalloc und tpfree entfallen
    - zusaetzliche Definition von ret_upic, upic_init, upic_call und
      upic_disable
      Die Source xattoupi.c enthaelt die Aufrufe upic_...
    - statt tpinit wird upic_init aufgerufen
      Auswertungen von tpurcode, tperrno entfallen. Stattdessen ret_upic
    - statt tpcall wird upic_call  aufgerufen
    - statt tpterm wird upic_disable aufgerufen
   DS 03.11.97
   - only upic call, no ifdef
   - no UTMBuffer, use msg.Data direct
*/

#define USE_UPIC_CALL
#include <windows.h>
#include <stdio.h>
#include <string.h>
#include <direct.h>
#include <process.h>
#include "xatmi.h" /* openUTM xatmi Header File */

#include "trans.h"
#include "pipe_routines.h"
#include "utm.h"

#define SERVICE_BUF_SIZE 16
#define MAX_TPP 40            // max treads per process
```

```
typedef char *EXTENSION_CONROL_BLOCK;
const int   TIMEOUT= 1000*30; // timeout in milliseconds
const int   ARGSIZE= 1024;
const char  *LOG_PATH="c:\\temp\\utm_logs\\";
const char  *LOG_NAME="client_%d.txt";

// Global variables set as parameters
extern char local_name[8];      // global for one Process

TPCLTINFO   client_info;
BOOL        bDone;      // executable termination request flag

static SECURITY_ATTRIBUTES sa;
static PSECURITY_DESCRIPTOR pSD;

static void __cdecl MainThread( void *ptr );

DWORD           dwMasterUtm = 0;
DWORD           dwAbortFlag = FALSE ;
HANDLE           hUtmMem = NULL ;
UTM_SHARED_MEM *lpUtmMem = NULL ;
HANDLE          evTerminate ;
HANDLE          evUtmMemInit ;
HANDLE          smBreak ;
DWORD           ProcessNumber ;

BOOL UTMTransaction(DWORD dwId, char *Service, void *Data)
{
       int            ret_upic;
       int            sendlen = sizeof(UTM_DATA);
       int            reclen = 0;

       Trace("about call utm-service %s\n", Service);

    if ( (ret_upic = upic_call(dwId, Service,
                                                          (char *)Data,
sendlen,
                                                          (char *)Data,
&reclen)) != 0 )
    {
        Trace( "UTMTransaction: upic_call failed, ret_upic=%d\n",
ret_upic);
        return FALSE;
    }

       Trace( "utm call retuned %d bytes\n", reclen);

    if (reclen < sendlen)
    {
        Trace( "UTMTransaction: reclen(%d) < sendlen(%d)\n",
                          reclen,sendlen );
        return FALSE;
    }
    return TRUE;
}


BOOL HandleTransactions(DWORD dwId, SM_PIPE *pPipe)
{
        UTM_MSG msg;
        DWORD nRead;

        while(ReadPipe(pPipe, &msg, sizeof(msg),&nRead))
        {
                DWORD nWritten;

                if(!UTMTransaction(dwId, msg.Service, &msg.Data))
                {
                        Trace( "UTMTransaction failed\n");
                        if (!msg.Data.Error)
                        {
                                // let front end know, that we have a
problem here
                                msg.Data.Error = TRUE;
                                strcpy
(msg.Data.Trans.ErrorMsg,"UTMTransaction failed");
                        }
                }

                if(!WritePipe(pPipe, &msg, nRead, &nWritten))
                {
                        Trace(" WritePipe Failed in
HandleTransactions()\n");
                        // can't inform front end without write !
                        return(FALSE) ;
                }

                if(nWritten != nRead)
                {
                        Trace( "HandleTransactions: nWritten(%d) !=
nRead(%d)\n", nWritten, nRead);
                }
        }

        return(TRUE) ;
}


void Abort(SM_PIPE *pPipe)
{
        dwAbortFlag = TRUE ;

        if(pPipe->evRDav)
                                        CloseHandle(pPipe->evRDav) ;
        if(pPipe->evWDav)
                                        CloseHandle(pPipe->evWDav) ;
}


/* FUNCTION: void MainThread( void *ptr )
 *
 * PURPOSE: This function is executed inside the client threads.
 *
 * ARGUMENTS:   void    *ptr    dummy argument passed in though thread
manager, unused NULL.
 *
 * RETURNS:     None
 *
 * COMMENTS:    will be identified by global int ClientNumber
 *
 */
static void __cdecl MainThread( void *ptr )
{
        SM_PIPE   Pipe;
        DWORD   dwId;      // this is the connection id
        int     ret_upic;

        dwId = (DWORD) ptr ;

        Pipe.evRDav = Pipe.evWDav = NULL ;

        if(OpenServerPipe(&Pipe, dwId, &sa, lpUtmMem) == FALSE)
        {
                Trace( "Thread %d - OpenServerPipe failed\n", dwId) ;

                Abort(&Pipe) ;

                return;
        }

        Pipe.hStop = evTerminate ;

        Trace( "Thread %d - open pipe ok\n", dwId) ;

        if ( (ret_upic = upic_init()) != 0 )
        {
                Trace("\nAbnormal termination of ret_upic\n"
                             "ret_upic: %d \n" , ret_upic );

                Abort(&Pipe) ;
                return;
        }

        InterlockedIncrement(&lpUtmMem->lConnections) ;

        if(HandleTransactions(dwId, &Pipe))
```

```
                                        dwAbortFlag = TRUE ;

        if(Pipe.evRDav)
                                        CloseHandle(Pipe.evRDav) ;

        if(Pipe.evWDav)
                                        CloseHandle(Pipe.evWDav) ;

        (void)upic_disable();

        InterlockedDecrement(&lpUtmMem->lConnections) ;

        return ;
}


int CreatePipeMem(DWORD dwConnections)
{
        hUtmMem = CreateFileMapping((HANDLE)0xFFFFFFFF, &sa,

PAGE_READWRITE | SEC_COMMIT, 0,

dwConnections*(sizeof(UTM_MSG)+sizeof(DWORD)+sizeof(UTM_HANDLES)),
                                                              UTM_MEM_SPACE
);

        if(!hUtmMem)
        {
                Trace("0x%x: Can not create pipe-shared memory\n",
GetLastError()) ;

                return(2) ;
        }

        if(GetLastError() == ERROR_ALREADY_EXISTS)
        {
                Trace("Another process is the UTM-Master\n",
GetLastError()) ;

                return(2) ;
        }

        lpUtmMem = MapViewOfFile(hUtmMem, FILE_MAP_ALL_ACCESS, 0, 0, 0) ;

        if(!lpUtmMem)
        {
                Trace("0x%x: Can not map pipe-shared memory\n",
GetLastError()) ;

                return(0) ;
        }

        lpUtmMem->dwMaxConnections = dwConnections ;
        lpUtmMem->lConnections      = 0 ;
        lpUtmMem->dwCpp             = MAX_TPP ;
        lpUtmMem->dwMaxTransferLen = sizeof(UTM_MSG) ;
        lpUtmMem->dwPIdMasterUtm   = GetCurrentProcessId() ;
        lpUtmMem->evTerminate      = evTerminate = CreateEvent(NULL,
TRUE, FALSE, NULL) ;
        lpUtmMem->smBreak          = smBreak = CreateSemaphore(&sa, 250,
250, NULL) ;

        if(!evTerminate || !smBreak)
        {
                Trace("0x%x: Can not create termination event\n",
GetLastError()) ;

                return(0) ;
        }

        return(1) ;
}



BOOL OpenPipeMem()
{
        hUtmMem = OpenFileMapping(FILE_MAP_ALL_ACCESS, FALSE,
UTM_MEM_SPACE) ;

        if(hUtmMem == NULL)
        {
                Trace("Can not open pipe-shared memory\n",
GetLastError()) ;
                return(FALSE) ;
        }

        lpUtmMem = MapViewOfFile(hUtmMem, FILE_MAP_ALL_ACCESS, 0, 0, 0) ;

        if(lpUtmMem)
        {
                evTerminate = DuplicateUtmHandle(lpUtmMem->evTerminate,
lpUtmMem->dwPIdMasterUtm) ;

                if(evTerminate)
                {
                        smBreak = DuplicateUtmHandle(lpUtmMem->smBreak,
lpUtmMem->dwPIdMasterUtm) ;

                        if(smBreak)
                                        return(TRUE) ;

                        Trace("0x%x: Can not duplicate termination
event\n", GetLastError()) ;
                } else Trace("0x%x: Can not duplicate termination
event\n", GetLastError()) ;
```

```
        }
        else Trace("0x%x: Can not map pipe-shared memory\n",
GetLastError()) ;

        return(FALSE) ;
}



BOOL __stdcall CtrlHandler(DWORD dwCtrlType)
{
        switch(dwCtrlType)
         {
          case CTRL_C_EVENT:
          case CTRL_BREAK_EVENT:
          case CTRL_CLOSE_EVENT:
          case CTRL_SHUTDOWN_EVENT:

                      Trace("Abort in process....\n") ;
                      SetEvent(evTerminate) ;

                      return(TRUE) ;
         }

        return(FALSE) ;
}


void CleanUp()
{
        if(evTerminate)
                                     SetEvent(evTerminate) ;

        if(lpUtmMem)
        {
                while(lpUtmMem->lConnections)
                              Sleep(100) ;

                UnmapViewOfFile(lpUtmMem) ;
        }

        if(hUtmMem)
                              CloseHandle(hUtmMem) ;

        if(evTerminate)
                              CloseHandle(evTerminate) ;

        if(dwMasterUtm)
                              CloseHandle(evUtmMemInit) ;

        if(smBreak)
                              CloseHandle(smBreak) ;
}
```

```

int __cdecl main(int argc, char ** argv)
{
        int iRepeat;
        int iPipeCount ;

        if (argc != 2)
        {
                fprintf(stderr, "usage: %s <remaining number of
pipes>\n", argv[0]);
                exit(1);
        }

        iPipeCount = atoi(argv[1]);

        if(iPipeCount < 1)
        {
                fprintf(stderr, "Bad number of remaining pipes\n") ;
                exit(1) ;
        }

        ProcessNumber = iPipeCount / MAX_TPP ;

#ifdef _DEBUG
        {
         char buf[_MAX_PATH];

         strcpy(buf, LOG_PATH);
         _mkdir(LOG_PATH);
         sprintf(buf+strlen(buf), LOG_NAME, ProcessNumber);
         freopen(buf, "w", stderr);

         setbuf(stderr, NULL);
        }
#endif

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

        Trace("utmclient %d starting with remaning pipes %d (as thread
0x%x)\n",
                  ProcessNumber, iPipeCount, GetCurrentThreadId());

        // general for all threads of this process
        strcpy ( local_name, "schwarz" );
        strcpy ( client_info.cltname, "schwarz" );
        strcpy ( client_info.usrname, "" );
        strcpy ( client_info.passwd, "" );

        switch(CreatePipeMem((DWORD) iPipeCount))
        {
         case 0: // Fatal error during shared mem init.

                  CleanUp() ;
                  exit(1) ;

         case 1: // The process is the Utm-Master

                  dwMasterUtm  = 1 ;
                  evUtmMemInit = CreateEvent(&sa, TRUE, FALSE,
UTM_MEM_EVENT) ;
                  break ;

         case 2: // Another process is the Utm-Master

                  if(OpenPipeMem() == FALSE)
                  {
                      CleanUp() ;
                      exit(1) ;
                  }
        }

        if(!dwMasterUtm)
        {
                iRepeat = iPipeCount > MAX_TPP ? MAX_TPP

        : iPipeCount ;

                while(iRepeat--)
                {
                        iPipeCount-- ;
```

```
                         // Start the child thread
                         if(_beginthread(MainThread, 0, (void *)
iPipeCount) == -1 )
                         {
                                 Trace( "Unable to start another thread,
number=%d\n", iPipeCount);
                                 exit (1);
                         }
                }
        }

        SetConsoleCtrlHandler(CtrlHandler, TRUE) ;

        if(iPipeCount)
        {
                STARTUPINFO         StartupInfo ;
                PROCESS_INFORMATION ProcessInformation ;
                char CmdLine[_MAX_PATH+20] ;

                wsprintf(CmdLine, "%s %d", argv[0], iPipeCount) ;

                GetStartupInfo(&StartupInfo) ;

                if(CreateProcess(argv[0], CmdLine, &sa, &sa, FALSE,
                                               NORMAL_PRIORITY_CLASS,
                                               NULL, NULL, &StartupInfo,
&ProcessInformation))
                {
                          CloseHandle(ProcessInformation.hProcess) ;
                          CloseHandle(ProcessInformation.hThread) ;
                }
                else
                {
                             CleanUp() ;
                             exit(1) ;
                }
        }

        if(dwMasterUtm)
        {
                while((DWORD) lpUtmMem->lConnections != lpUtmMem-
>dwMaxConnections &&   !dwAbortFlag)
                             Sleep(100) ;

                if(!dwAbortFlag)
                             SetEvent(evUtmMemInit) ;
        }

        WaitForSingleObject(evTerminate, INFINITE) ;

        CleanUp() ;
        return(dwAbortFlag == TRUE ? 1 : 0) ;
}

#define  UPICL_WIN32
#define  UTM_ON_WIN32
#include  <windows.h>
#include  <upic.h>
#include  <stdio.h>
#include  "pipe_routines.h"
#include  "trans.h"
#include  "utm.h"

#define LogFile stderr
// #define MAX_RECLEN 4096
#define MAX_RECLEN sizeof(UTM_DATA)

char    local_name[9];       // global for one Process

extern int ProcessNumber;

/* -------------------------------------- upic_init() --------------------
-----------*/
int upic_init()
{
        long            local_name_lth;
        CM_RETCODE      return_code;

        local_name_lth = strlen(local_name);
        Enable_UTM_UPIC ( (unsigned char *)local_name, &local_name_lth,
&return_code ) ;
        if ( return_code   !=  CM_OK )
        {
                fprintf (LogFile,"*** Enable_UTM_UPIC(): error %d\n",
return_code );
        }
        return (return_code);
}

/* --------------------------------------- upic_disable() ----------------
---------------*/
int upic_disable()
{
        CM_RETCODE      return_code;
        long            local_name_lth;

        local_name_lth = strlen(local_name);
        Disable_UTM_UPIC ( (unsigned char *)local_name, &local_name_lth,
&return_code );
        return (0);
}

/* ------------------------------------- upic_call() ---------------------
-----------*/
```

```
int upic_call(DWORD dwId, char *service, char *sendbuff, int sendlen, char
*recbuff, int *reclen)
{

        long                    local_name_lth=8;
        CONVERSATION_ID         upic_conv_ID1;
        CM_RETCODE              return_code;
        DATA_RECEIVED  data_rcv;
        STATUS_RECEIVED         status_rcv;
        REQUEST_TO_SEND_RECEIVED   rq_to_send_rcv;
        unsigned char  sym_dest_name[9] = "SAMPLE00";
        long                    sym_dest_name_lth = 8;
        unsigned char  tp_name[9];
        long                    tp_name_lth = 0;
        long                    requ_lth;
        long                    rcv_lth;

        switch (  service[0] )
        {
                case ’N’:
                        strcpy( tp_name, "KNORDER");
                        tp_name_lth = 7;
                    break;
                case ’S’:
                        strcpy( tp_name, "KSTOCKL");
                        tp_name_lth = 7;
             break;
                case ’P’:
                        strcpy( tp_name, "KPAYMENT");
                        tp_name_lth = 8;
             break;
                case ’O’:
                        strcpy( tp_name, "KORDERST");
                        tp_name_lth = 8;
             break;
                default :
                        Trace("unknown service %s \n",service);
                        return (-99);
        }  /* end switch (service[0] */

        sprintf(sym_dest_name, "SERV10%02d", ProcessNumber+1);

        /* Initialize_Conversation - Call */

        Initialize_Conversation ( upic_conv_ID1, sym_dest_name,
&return_code );
        if ( return_code != CM_OK )
    {
                fprintf ( LogFile,"*** Initialize_Conversation() %s: error
%d\n", sym_dest_name, return_code );
                upic_disable();
                return (return_code);
        }
```

```
       /* Set_TP_Name - Call */

    Set_TP_Name ( upic_conv_ID1 , tp_name , &tp_name_lth , &return_code );
    if ( return_code != CM_OK )
    {
               fprintf ( LogFile,"*** Set_TP_Name(): error %d\n",
return_code );
               upic_disable();
               return (return_code);
    }

    /* Allocate - Call */

       {
                int iI = 0 ;

                while(1)
                {
                         Allocate ( upic_conv_ID1, &return_code );

                         if ( return_code  !=  CM_OK )
                         {
                                  fprintf ( LogFile,"*** Allocate(%d):
error %d\n", dwId, return_code );

                                  if(++iI == 10)
                                              return (return_code);
                                  else Sleep(250) ;
                         }
                         else break ;
                }
       }

       Send_Data (    upic_conv_ID1,
               (unsigned char *) sendbuff,
               &sendlen,
               &rq_to_send_rcv,
               &return_code
               );
       if ( return_code != CM_OK )
    {
               fprintf ( LogFile, "*** Send_Data(): error %d\n",
return_code );
               upic_disable();
               return (return_code);
       }

       /* 1. Receive - Call for Data  */
    requ_lth = MAX_RECLEN;
    Receive ( upic_conv_ID1,
               (unsigned char *) recbuff,
               &requ_lth,
               &data_rcv,
               &rcv_lth,
               &status_rcv,
               &rq_to_send_rcv,
               &return_code
               );
       if ( ( return_code  ==  CM_OK )  ||
          ( return_code  ==  CM_DEALLOCATED_NORMAL ) )
       {
               if ( data_rcv  !=  CM_NO_DATA_RECEIVED )
            *reclen = rcv_lth;
       }
       else
       {
               fprintf ( LogFile,"*** 1. Receive(): error %d\n",
return_code );
               upic_disable();
               return (return_code);
       }

       /* 2. Receive - Call for Status CM_DEALLOCATED_NORMAL */
    if (  return_code  == CM_OK )
       {
               requ_lth = 0;
               Receive ( upic_conv_ID1,
                      (unsigned char *) recbuff,
                      &requ_lth,
                      &data_rcv,
                      &rcv_lth,
                      &status_rcv,
                      &rq_to_send_rcv,
                      &return_code
                      );
               if ( return_code  !=  CM_DEALLOCATED_NORMAL  )
        {
                      fprintf ( LogFile,"*** 2. Receive(): error %d\n",
return_code );
                      upic_disable();
                      return (return_code);
               }
       }

       return (0);
}

# Microsoft Developer Studio Generated NMAKE File, Format Version 4.10
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Console Application" 0x0103

!IF "$(CFG)" == ""
CFG=utm_client - Win32 Debug
```

```
!MESSAGE No configuration specified.  Defaulting to utm_client - Win32
Debug.
!ENDIF

!IF "$(CFG)" != "utm_client - Win32 Release" && "$(CFG)" !=\
 "utm_client - Win32 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 /f "utm_client.mak" CFG="utm_client - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "utm_client - Win32 Release" (based on\
 "Win32 (x86) Console Application")
!MESSAGE "utm_client - Win32 Debug" (based on\
 "Win32 (x86) Console Application")
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF

!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
##########################################################################
######
# Begin Project
# PROP Target_Last_Scanned "utm_client - Win32 Debug"
CPP=cl.exe
RSC=rc.exe

!IF  "$(CFG)" == "utm_client - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
OUTDIR=.\Release
INTDIR=.\Release

ALL : "$(OUTDIR)\utm_client.exe"

CLEAN :
       -@erase "$(INTDIR)\pipe_routines.obj"
       -@erase "$(INTDIR)\utm_client.obj"
       -@erase "$(INTDIR)\XATTOUPI.OBJ"
       -@erase "$(OUTDIR)\utm_client.exe"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE"
/YX /c
# ADD CPP /nologo /MT /W3 /GX /O2 /D "NDEBUG" /D "WIN32" /D "_CONSOLE" /YX
/c
CPP_PROJ=/nologo /MT /W3 /GX /O2 /D "NDEBUG" /D "WIN32" /D "_CONSOLE"\
 /Fp"$(INTDIR)/utm_client.pch" /YX /Fo"$(INTDIR)/" /c
CPP_OBJS=.\Release/
CPP_SBRS=.\.
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/utm_client.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib upicw32.lib /nologo /subsystem:console /machine:I386
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib upicw32.lib /nologo /subsystem:console /incremental:no\
 /pdb:"$(OUTDIR)/utm_client.pdb" /machine:I386
/out:"$(OUTDIR)/utm_client.exe"
LINK32_OBJS= \
       "$(INTDIR)\pipe_routines.obj" \
       "$(INTDIR)\utm_client.obj" \
       "$(INTDIR)\XATTOUPI.OBJ"

"$(OUTDIR)\utm_client.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ELSEIF  "$(CFG)" == "utm_client - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
OUTDIR=.\Debug
INTDIR=.\Debug

ALL : "$(OUTDIR)\utm_client.exe"

CLEAN :
        -@erase "$(INTDIR)\pipe_routines.obj"
        -@erase "$(INTDIR)\utm_client.obj"
        -@erase "$(INTDIR)\vc40.idb"
        -@erase "$(INTDIR)\vc40.pdb"
        -@erase "$(INTDIR)\XATTOUPI.OBJ"
        -@erase "$(OUTDIR)\utm_client.exe"
        -@erase "$(OUTDIR)\utm_client.ilk"
        -@erase "$(OUTDIR)\utm_client.pdb"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_CONSOLE" /YX /c
# ADD CPP /nologo /MT /W3 /Gm /GX /Zi /Od /D "_DEBUG" /D "WIN32" /D
"_CONSOLE" /YX /c
CPP_PROJ=/nologo /MT /W3 /Gm /GX /Zi /Od /D "_DEBUG" /D "WIN32" /D
"_CONSOLE"\
 /Fp"$(INTDIR)/utm_client.pch" /YX /Fo"$(INTDIR)/" /Fd"$(INTDIR)/" /c
CPP_OBJS=.\Debug/
CPP_SBRS=.\.
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/utm_client.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /debug /machine:I386
# ADD LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib
advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib
odbccp32.lib upicw32.lib /nologo /subsystem:console /debug /machine:I386
LINK32_FLAGS=kernel32.lib user32.lib gdi32.lib winspool.lib comdlg32.lib\
 advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib odbc32.lib\
 odbccp32.lib upicw32.lib /nologo /subsystem:console /incremental:yes\
 /pdb:"$(OUTDIR)/utm_client.pdb" /debug /machine:I386\
 /out:"$(OUTDIR)/utm_client.exe"
LINK32_OBJS= \
```

```
        "$(INTDIR)\pipe_routines.obj" \
        "$(INTDIR)\utm_client.obj" \
        "$(INTDIR)\XATTOUPI.OBJ"

"$(OUTDIR)\utm_client.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ENDIF

.c{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.c{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

##########################################################################
######
# Begin Target

# Name "utm_client - Win32 Release"
# Name "utm_client - Win32 Debug"

!IF  "$(CFG)" == "utm_client - Win32 Release"

!ELSEIF  "$(CFG)" == "utm_client - Win32 Debug"

!ENDIF

##########################################################################
######
# Begin Source File

SOURCE=.\XATTOUPI.C
DEP_CPP_XATTO=\
        {$(INCLUDE)}"\pipe_routines.h"\
        {$(INCLUDE)}"\upic.h"\


"$(INTDIR)\XATTOUPI.OBJ" : $(SOURCE) $(DEP_CPP_XATTO) "$(INTDIR)"
```

```
# End Source File
########################################################################
######
# Begin Source File

SOURCE=.\utm_client.c
DEP_CPP_UTM_C=\
	{$(INCLUDE)}"\pipe_routines.h"\
	{$(INCLUDE)}"\sqldb.h"\
	{$(INCLUDE)}"\sqlfront.h"\
	{$(INCLUDE)}"\trans.h"\
	{$(INCLUDE)}"\utm.h"\
	{$(INCLUDE)}"\xatmi.h"\
	{$(INCLUDE)}"\xatmidef.h"\
	

"$(INTDIR)\utm_client.obj" : $(SOURCE) $(DEP_CPP_UTM_C) "$(INTDIR)"


# End Source File
```

Server Application Source Code

```
/*  ROOT SOURCE FOR APPLICATION SERV1  */

#define KDCENTRYNAME      kcxmnt
#define KDCUTMVERS        1
#define KDCMSGFILE        msgpriv
#define KDCMSGFILENAME    {’m’,’s’,’g’,’p’,’r’,’i’,’v’,’ ’}
#define KDCVERSION        {’4’,’.’,’0’,’A’}
#define KDCDEFTIME        25478
#define KDCLTHKBPRG       1
#define KDCLTHSPAB        1000
#define KDCLTHMPUTAREA    4096
#define KDCLTHFMIOAREA    4120
#define KDCLTHRESTART     20480
#define KDCCLEARCH        0XAF
#if defined (__STDC__) && (__STDC__ == 1)
void KCSTRMA (char *) ;
#else
void KCSTRMA () ;
#endif
#define KDCFH             0
void f_formcon ()  { KCSTRMA ("NOFORM") ; }
static  char    korrver [9] = "NONE    ";
char *  n_korrver = &korrver[0];
#define KDCSTRTEXIT       1
#define KDCSHUTEXIT       1
```

```
########################################################################
######
# Begin Source File

SOURCE="\openUTM-SRC\AUDIT\shared\pipe_routines.c"
DEP_CPP_PIPE_=\
	{$(INCLUDE)}"\pipe_routines.h"\
	{$(INCLUDE)}"\sqldb.h"\
	{$(INCLUDE)}"\sqlfront.h"\
	{$(INCLUDE)}"\trans.h"\
	{$(INCLUDE)}"\utm.h"\
	

"$(INTDIR)\pipe_routines.obj" : $(SOURCE) $(DEP_CPP_PIPE_) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)


# End Source File
# End Target
# End Project
########################################################################
######

 #define KDCINPUTFORM      0
 #define KDCINPUTLINE      0
 #define KDCINPUTUSER      0
 #define KDCBADTACS        0
 #define KDCMSGTAC         0
 #define KDCSIGNON         0
 #define KDCNRDB           0
 #define KDCLTHTAMifx      0
 #define KDCLTHTSKMifx     0
 #define KDCDBTYPE         0
 #define KDCLTHDBTAB       0
 #define KDCADRROOTTAM     (char *)&roottam
 #define KDCADRROOTTSKM    (char *)(-1)
 #define KDCADRDBOPCODE    (char *)(-1)
 #define KDCADRDBCONPAA    (char *)(-1)
 #define KDCADRDBENTRS     (char *)(-1)
 #define KDCADRDBCODING    (char *)(-1)
 #define KDCADRDBCODES     (char *)(-1)
 #define KDCADRTAMHDRSV    (char *)(-1)
 #define KDCADRDBTRACAR    (char *)(-1)
 #define KDCADRDBERRMSG    (char *)0
 #define KDCADRACNT        (char *)(-1)

 #include <xirtstrt.h>


 static struct linksect
         {
         char *addr_kdckb;
         char *addr_kdcspab;
```

```
        int end;
        } linksect =
        {
        (char *)0, /* set by kcxrtst */
        (char *)0, /* set by kcxrtst */
        (-2)
        };

#define KDCLASTADRLSECT   (char *)(-1)

extern void KDCADM();
extern void svrinit();
extern void svrdone();
extern void KNEW_ORDER();
extern void KSTOCK_LEVEL();
extern void KPAYMENT();
extern void KORDER_STATUS();

#define KDC_BLSGEN        0
static struct sprgtabl
    {
    struct prdc1
        {
        char program_name[32];
        char *program_addr;
        } prdc1;
    struct prdc2
        {
        char program_name[32];
        char *program_addr;
        } prdc2;
    struct prdc3
        {
        char program_name[32];
        char *program_addr;
        } prdc3;
    struct prdc4
        {
        char program_name[32];
        char *program_addr;
        } prdc4;
    struct prdc5
        {
        char program_name[32];
        char *program_addr;
        } prdc5;
    struct prdc6
        {
        char program_name[32];
        char *program_addr;
        } prdc6;
    struct prdc7
        {
```

```
        char program_name[32];
        char *program_addr;
        } prdc7;
    int endmark;
    } sprgtabl =
/* struct sprgtabl {        */ {
/*  struct prdc1     {      */   {
/*   char program_name[32]; */   {’K’,’D’,’C’,’A’,’D’,’M’,’ ’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’},
/*   char *program_addr;    */   (char *)KDCADM,
/*   } prdc1    ;           */   },
/*  struct prdc2     {      */   {
/*   char program_name[32]; */   {’s’,’v’,’r’,’i’,’n’,’i’,’t’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’},
/*   char *program_addr;    */   (char *)svrinit,
/*   } prdc2    ;           */   },
/*  struct prdc3     {      */   {
/*   char program_name[32]; */   {’s’,’v’,’r’,’d’,’o’,’n’,’e’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’},
/*   char *program_addr;    */   (char *)svrdone,
/*   } prdc3    ;           */   },
/*  struct prdc4     {      */   {
/*   char program_name[32]; */
{’K’,’N’,’E’,’W’,’_’,’O’,’R’,’D’,’E’,’R’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’},
/*   char *program_addr;    */   (char *)KNEW_ORDER,
/*   } prdc4    ;           */   },
/*  struct prdc5     {      */   {
/*   char program_name[32]; */
{’K’,’S’,’T’,’O’,’C’,’K’,’_’,’L’,’E’,’V’,’E’,’L’,’ ’,’ ’,’ ’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’},
/*   char *program_addr;    */   (char *)KSTOCK_LEVEL,
/*   } prdc5    ;           */   },
/*  struct prdc6     {      */   {
/*   char program_name[32]; */   {’K’,’P’,’A’,’Y’,’M’,’E’,’N’,’T’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’},
/*   char *program_addr;    */   (char *)KPAYMENT,
/*   } prdc6    ;           */   },
/*  struct prdc7     {      */   {
/*   char program_name[32]; */
{’K’,’O’,’R’,’D’,’E’,’R’,’_’,’S’,’T’,’A’,’T’,’U’,’S’,’ ’,’ ’,’ ’,’ ’,’
’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’,’ ’},
/*   char *program_addr;    */   (char *)KORDER_STATUS,
/*   } prdc7    ;           */   },
/* int endmark;             */ (-2)
/* } sprgtabl;              */ };

#define KDCNRPRG          7
```

```
#define KDCNRAREA         0

static short exindlst[] = {
          2
          ,          3
          ,          0
          ,          0
          ,          0
          ,          0
          ,          0
          ,          0
          ,          0
          };

#define KDCCOBCON         (void(*)())(-2)
#define KDCCOB2CON        (void(*)())(-2)

void KDCCC   ();
#define KDCCCON           KDCCC

#include <xirtend.h>

#include <xirtcc.h>
                         ((char **)(iutmhlpar->area_addr))[0],
                         ((char **)(iutmhlpar->area_addr))[1]
                         );
  }
#include <xirtcprt.h>
return;
}

#include <windows.h>
#include <stdio.h>
#include <time.h>
#include <stdarg.h>

// UTM include files
#include <xatmi.h>
#include <kcmac.h>

// Database include files
#define DBNTWIN32
#include <sqlfront.h>
#include <sqldb.h>

// include files for this project
#define UTM_SERVER
#include "trans.h"
#include "tpcc.h"
#include "sqlroutines.h"
#include "utm.h"
#include "error.h"
```

```
#ifdef _DEBUG
__inline void __cdecl Trace(PSTR pFormat, ...)
{
 va_list Parameter ;

 va_start(Parameter, pFormat) ;

 vfprintf(stderr, pFormat, Parameter) ;
}
#else
__inline void __cdecl Trace(PSTR pFormat, ...) {}
#endif

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


// defines fuer KDCS-Programm
#define SPACE " "
#define KBKOPF kb->kopf
#define KBRFLD kb->rfld
#define pb spab->call_pb

// Global variables
short   iMaxConnections= 1;
char    szErrorLogPath[]="\\inetpub\\wwwroot\\err_tpcc_utm.txt";
DBPROCESS *pdbproc;
char    *Server = NULL;;
char    *Database = "tpcc";
char    *User = "sa";
char    *Password = "";
int            spId;
UTM_DATA data;
// TERM Term;
extern char ErrorMsgBuffer[] ;

EXTENSION_CONTROL_BLOCK *gpECB = NULL;
CRITICAL_SECTION ErrorLogCriticalSection;
BOOL    SQL_CONNECTED = FALSE;

// structur for KDCS-Error
static struct s_errdaten
{
       char message[80];
    char kcrc[8];
} errdaten;
// structur for UTM-data
                                     /*  SPAB   */
static struct work
{
        struct kc_pa call_pb;
} *spab;
                                      /*  KB     */
static struct kc_ca
{
        struct ca_hdr kopf;
        struct ca_rti rfld;
        char   user[1];
} *kb;


void WriteZString(EXTENSION_CONTROL_BLOCK *pECB, char *szStr)
{
        strcpy(data.Trans.ErrorMsg, szStr);
        data.Error = 1;
}

BOOL IsValidTermId(int TermId)
{
        return FALSE;
}

/* 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
               error id
 *                            int                            oserr
               operating 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
 *
 */
int err_handler(DBPROCESS *dbproc, int severity, int dberr, int oserr,
char *dberrstr, char *oserrstr)
{
        PECBINFO                              pEcbInfo;
        EXTENSION_CONTROL_BLOCK       *pECB;
        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.", 0, 0);
                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 number
 *                            int                            msgstate
               message state
 *                            int                            severity
               message severity
 *                            char                  *msgtext
       printable message description
 *
 * RETURNS:           int                           INT_CONTINUE
       continue if error is SQLETIME else INT_CANCEL action
 *                                                           INT_CANCEL
               cancel operation
 *
 * COMMENTS:   This function also sets the dead lock dbproc variable if
necessary.
 *
 */
int msg_handler(DBPROCESS *dbproc, DBINT msgno, int msgstate, int
severity, char *msgtext)
{
       PECBINFO                              pEcbInfo;
       EXTENSION_CONTROL_BLOCK       *pECB;
       FILE                                  *fp;
       SYSTEMTIME                            systemTime;
       char                                  szTmp[256];
       int                                          iTermId;
       int                                          iSyncId;

       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;
}
```

```
BOOL SQLInit(void)
{
	extern short iMaxConnections;

	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);

	InitializeCriticalSection(&ErrorLogCriticalSection);
	return TRUE;
}

/* 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
 *
 */


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(ECBINFO));
	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");
	dbsqlexec(*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");

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

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

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

		return FALSE;
}


int  svrinit(int argc, char *argv[])
{
	char App[1024];
	char *sysname;
	Trace("starting the UTM TPCC Server");
	if (getenv("COMPUTERNAME"))
	{
		sysname = strdup(getenv("COMPUTERNAME"));
		sprintf (App, "%s",sysname);
	}
	else
		strcpy(App, "TPCC");

	if (!SQLInit())
	{
		Trace("SQLInit failed");
		return -1;
	}
	if (getenv("SERVER"))
		Server = strdup(getenv("SERVER"));
	if (Server == NULL)
	{
		Trace("SERVER Environment variable not set");
		return -1;
	}
	if (SQLOpenConnection(NULL, 0, 0, &pdbproc, Server, Database,
User, Password, App, &spId))
	{
		Trace("SQLOpenconnection failed");
		// SQLCleanup();
		dbexit();
		return -1;
	}
    SQL_CONNECTED = TRUE;
	return 0;
}


void  svrdone(void)
{
	Trace("Shut down UTM-server");
	free(Server);
//	SQLCloseConnection(NULL, pdbproc);
	dbclose(pdbproc);
//	SQLCleanup();
	dbexit();
}

/* 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: 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 process id
 *                              STOCK_LEVEL_DATA               *pStockLevel
        stock level input / output data structure
 *                              short
        deadlock_retry retry count if deadlocked
 *
 * RETURNS:            BOOL    FALSE          if successfull
 *                                            TRUE           if deadlocked
 *
 * COMMENTS:   None
 *
 */

BOOL SQLStockLevel(EXTENSION_CONTROL_BLOCK *pECB, int iTermId, int
iSyncId, DBPROCESS *dbproc, STOCK_LEVEL_DATA *pStockLevel, short
deadlock_retry)
{
        int                     tryit;
        RETCODE         rc;
        char            printbuf[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))
                                {
                                        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
commited.");
                        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 retry count if deadlocked
 *
 * RETURNS:           int     TRUE    transaction committed
 *                                    FALSE   item number not valid
 *                                    -1             deadlock max retry
reached
 *
 *
 * COMMENTS:   None
 *
 */

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 (dbrpcinit(dbproc, tmpbuf, 0) == SUCCEED)
```

```
               {
                      dbrpcparam(dbproc, NULL, 0, SQLINT2, -1, -1, (BYTE
*) &pNewOrder->w_id);
                      dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE
*) &pNewOrder->d_id);
                      dbrpcparam(dbproc, NULL, 0, SQLINT4, -1, -1, (BYTE
*) &pNewOrder->c_id);
                      dbrpcparam(dbproc, NULL, 0, SQLINT1, -1, -1, (BYTE
*) &pNewOrder->o_ol_cnt);

                      pNewOrder->o_all_local = 1;
                      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, SQLINT1, -1, -1, (BYTE
*) &pNewOrder->o_all_local);

                      for (i = 0; i < pNewOrder->o_ol_cnt; i++)
                      {
                             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) == SUCCEED)
                      {
                             pNewOrder->total_amount=0;

                             // Get resutls 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, pData, dbdatlen(dbproc,
1));

       if(pData=dbdata(dbproc, 2))
```

```
        pNewOrder->Ol[i].ol_stock = (*(DBSMALLINT *) pData);

        if(pData=dbdata(dbproc, 3))

        UtilStrCpy(pNewOrder->Ol[i].ol_brand_generic, pData,
dbdatlen(dbproc, 3));

        if(pData=dbdata(dbproc, 4))

        pNewOrder->Ol[i].ol_i_price = (*(DBFLT8 *) pData);


        if(pData=dbdata(dbproc, 5))


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



                                                                pNewOrder-
>total_amount = pNewOrder->total_amount + pNewOrder->Ol[i].ol_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, 4));

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




        if(pData=dbdata(dbproc, 6))

        UtilStrCpy(pNewOrder->c_credit, pData, dbdatlen(dbproc, 6));

        if(pData=dbdata(dbproc, 7))
                                                        {
                                                                datetime =
*((DBDATETIME *) pData);

        dbdatecrack(dbproc, &pNewOrder->o_entry_d, &datetime);
                                                        }

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

```
                                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                 terminal id of browser
 *                              int
        iSyncId                 sync id of browser
 *                              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
 *
 */

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) == SUCCEED)
                {
                        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)
                        {
                                dbrpcparam(dbproc, NULL, 0, SQLCHAR, -1,
strlen(pPayment->c_last), pPayment->c_last);
                        }
                }
                if (dbrpcexec(dbproc) == SUCCEED)
                {
                        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 commited.");
                        return TRUE;
                }
        }
        // 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
 *                                      0             No orders found for
customer
 *                                      1             Transaction
successfull
 *
 * COMMENTS:   None
 *
 */

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 (dbrpcinit(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 (dbrpcexec(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-
>OlOrderStatusData[i].ol_supply_w_id = (*(DBSMALLINT *) pData);
                                                if(pData=dbdata(dbproc, 2))
                                                        pOrderStatus-
>OlOrderStatusData[i].ol_i_id = (*(DBINT *) pData);
                                                if(pData=dbdata(dbproc, 3))
                                                        pOrderStatus-
>OlOrderStatusData[i].ol_quantity = (*(DBSMALLINT *) pData);
                                                if(pData=dbdata(dbproc, 4))
                                                        pOrderStatus-
>OlOrderStatusData[i].ol_amount = (*(DBFLT8 *) pData);
```

```
                                                if(pData=dbdata(dbproc, 5))
                                                {
                                                        datetime =
*((DBDATETIME *) pData);
                                                        dbdatecrack(dbproc,
&pOrderStatus->OlOrderStatusData[i].ol_delivery_d, &datetime);
                                                }
                                                i++;
                                        }
                                        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
                                strcpy(pOrderStatus-
>execution_status,"Transaction commited.");
                        return 1;
                }
        }
        // If we reached here, it means we quit after MAX_RETRY deadlocks
        strcpy(pOrderStatus->execution_status,"Hit deadlock max.  ");
        return -1; //"deadlock max retry reached!"
}


PECBINFO SQLGetECB(PDBPROCESS p)
{
        return (PECBINFO) dbgetuserdata(p);
}


// Transact NEW_ORDER
void KNEW_ORDER(struct kc_ca *x_kb, struct work *x_spab)
{
        PECBINFO pECBInfo;
        int size;
        kb = x_kb;
        spab = x_spab;

    pb.kcop[0] = 'I';
        pb.kcop[1] = 'N';
        pb.kcop[2] = 'I';
        pb.kcop[3] = 'T';
        pb.kclcapa = 0;
        pb.kclspa  = sizeof(struct work);
        KDCS (&pb);

        // read data - length in KBRFLD.kcrlm
        pb.kcop[0] = 'M';
        pb.kcop[1] = 'G';
        pb.kcop[2] = 'E';
        pb.kcop[3] = 'T';
        pb.kcla = sizeof(data);
        pb.kcfn[0] = ’ ’;pb.kcfn[1] = ’ ’; pb.kcfn[2] = ’ ’; pb.kcfn[3] =
’ ’;
        pb.kcfn[4] = ’ ’;pb.kcfn[5] = ’ ’; pb.kcfn[6] = ’ ’; pb.kcfn[7] =
’ ’;
        KDCS( &pb, &data);

        pECBInfo = SQLGetECB(pdbproc);
    size = KBRFLD.kcrlm;

        Trace("Beginning NEW_ORDER transaction\n");

        data.Error = 0;
        data.Return = SQLNewOrder(NULL, data.TermId, data.SyncId, pdbproc,
                                            &data.Trans.NewOrderData,
data.DeadlockRetry);
        data.bDeadlock = pECBInfo->bDeadlock;
        data.bFailed = pECBInfo->bFailed;
        if (data.Error)
        {
                strcpy(data.Trans.ErrorMsg, ErrorMsgBuffer);
        }

        Trace("Finished NEWORDER transaction, bFailed=%d\n",
data.bFailed);

        pb.kcop[0] = ’M’;
        pb.kcop[1] = ’P’;
        pb.kcop[2] = ’U’;
        pb.kcop[3] = ’T’;
        pb.kcom[0] = ’N’;
        pb.kcom[1] = ’T’;
        pb.kcfn[0] = ’ ’;pb.kcfn[1] = ’ ’; pb.kcfn[2] = ’ ’; pb.kcfn[3] =
’ ’;
        pb.kcfn[4] = ’ ’;pb.kcfn[5] = ’ ’; pb.kcfn[6] = ’ ’; pb.kcfn[7] =
’ ’;
        pb.kcrn[0] = ’ ’;pb.kcrn[1] = ’ ’; pb.kcrn[2] = ’ ’; pb.kcrn[3] =
’ ’;
        pb.kcrn[4] = ’ ’;pb.kcrn[5] = ’ ’; pb.kcrn[6] = ’ ’; pb.kcrn[7] =
’ ’;
        pb.kcdf = 0;
        pb.kclm = size;
        KDCS(&pb, &data);

        pb.kcop[0] = ’P’;
        pb.kcop[1] = ’E’;
        pb.kcop[2] = ’N’;
        pb.kcop[3] = ’D’;
        pb.kcom[0] = ’F’;
        pb.kcom[1] = ’I’;
        KDCS(&pb);
}
```

```
// Transact STOCK_LEVEL
void KSTOCK_LEVEL(struct kc_ca *x_kb, struct work *x_spab)
{
        PECBINFO pECBInfo;
        int size;
        kb = x_kb;
        spab = x_spab;

    pb.kcop[0] = ’I’;
        pb.kcop[1] = ’N’;
        pb.kcop[2] = ’I’;
        pb.kcop[3] = ’T’;
        pb.kclcapa = 0;
        pb.kclspa  = sizeof(struct work);
        KDCS (&pb);

        // read data - length in KBRFLD.kcrlm
        pb.kcop[0] = ’M’;
        pb.kcop[1] = ’G’;
        pb.kcop[2] = ’E’;
        pb.kcop[3] = ’T’;
        pb.kcla = sizeof(data);
        pb.kcfn[0] = ’ ’;pb.kcfn[1] = ’ ’; pb.kcfn[2] = ’ ’; pb.kcfn[3] =
’ ’;
        pb.kcfn[4] = ’ ’;pb.kcfn[5] = ’ ’; pb.kcfn[6] = ’ ’; pb.kcfn[7] =
’ ’;
        KDCS( &pb, &data);

        pECBInfo = SQLGetECB(pdbproc);
        size = KBRFLD.kcrlm;

        Trace("Beginning STOCK_ LEVEL transaction\n");

        data.Error = 0;
        data.Return = SQLStockLevel(NULL, data.TermId, data.SyncId,
pdbproc,
                                              &data.Trans.StockLevelData,
data.DeadlockRetry);
        data.bDeadlock = pECBInfo->bDeadlock;
        data.bFailed = pECBInfo->bFailed;
        if (data.Error)
        {
                strcpy(data.Trans.ErrorMsg, ErrorMsgBuffer);
        }

        Trace("Finished STOCK_ LEVEL transaction, bFailed=%d\n",
data.bFailed);

        pb.kcop[0] = ’M’;
        pb.kcop[1] = ’P’;
        pb.kcop[2] = ’U’;
        pb.kcop[3] = ’T’;
        pb.kcom[0] = ’N’;
        pb.kcom[1] = ’T’;
        pb.kcfn[0] = ’ ’;pb.kcfn[1] = ’ ’; pb.kcfn[2] = ’ ’; pb.kcfn[3] =
’ ’;
        pb.kcfn[4] = ’ ’;pb.kcfn[5] = ’ ’; pb.kcfn[6] = ’ ’; pb.kcfn[7] =
’ ’;
        pb.kcrn[0] = ’ ’;pb.kcrn[1] = ’ ’; pb.kcrn[2] = ’ ’; pb.kcrn[3] =
’ ’;
        pb.kcrn[4] = ’ ’;pb.kcrn[5] = ’ ’; pb.kcrn[6] = ’ ’; pb.kcrn[7] =
’ ’;
        pb.kcdf = 0;
        pb.kclm = size;
        KDCS(&pb, &data);

        pb.kcop[0] = ’P’;
        pb.kcop[1] = ’E’;
        pb.kcop[2] = ’N’;
        pb.kcop[3] = ’D’;
        pb.kcom[0] = ’F’;
        pb.kcom[1] = ’I’;
        KDCS(&pb);
}


// Transact PAYMENT
void KPAYMENT(struct kc_ca *x_kb, struct work *x_spab)
{
        PECBINFO pECBInfo;
        int size;
        kb = x_kb;
    spab = x_spab;

    pb.kcop[0] = ’I’;
        pb.kcop[1] = ’N’;
        pb.kcop[2] = ’I’;
        pb.kcop[3] = ’T’;
        pb.kclcapa = 0;
        pb.kclspa  = sizeof(struct work);
        KDCS (&pb);

    // read data - length in KBRFLD.kcrlm
        pb.kcop[0] = ’M’;
        pb.kcop[1] = ’G’;
        pb.kcop[2] = ’E’;
        pb.kcop[3] = ’T’;
        pb.kcla = sizeof(data);
        pb.kcfn[0] = ’ ’;pb.kcfn[1] = ’ ’; pb.kcfn[2] = ’ ’; pb.kcfn[3] =
’ ’;
        pb.kcfn[4] = ’ ’;pb.kcfn[5] = ’ ’; pb.kcfn[6] = ’ ’; pb.kcfn[7] =
’ ’;
        KDCS( &pb, &data);

        pECBInfo = SQLGetECB(pdbproc);
        size = KBRFLD.kcrlm;
```

```

        Trace("Beginning PAYMENT transaction\n");

        data.Error = 0;
        data.Return = SQLPayment(NULL, data.TermId, data.SyncId, pdbproc,
                                             &data.Trans.PaymentData,
data.DeadlockRetry);
        data.bDeadlock = pECBInfo->bDeadlock;
        data.bFailed = pECBInfo->bFailed;
        if (data.Error)
        {
                strcpy(data.Trans.ErrorMsg, ErrorMsgBuffer);
        }

        Trace("Finished PAYMENT transaction\n");

        pb.kcop[0] = ’M’;
        pb.kcop[1] = ’P’;
        pb.kcop[2] = ’U’;
        pb.kcop[3] = ’T’;
        pb.kcom[0] = ’N’;
        pb.kcom[1] = ’T’;
        pb.kcfn[0] = ’ ’;pb.kcfn[1] = ’ ’; pb.kcfn[2] = ’ ’; pb.kcfn[3] =
’ ’;
        pb.kcfn[4] = ’ ’;pb.kcfn[5] = ’ ’; pb.kcfn[6] = ’ ’; pb.kcfn[7] =
’ ’;
        pb.kcrn[0] = ’ ’;pb.kcrn[1] = ’ ’; pb.kcrn[2] = ’ ’; pb.kcrn[3] =
’ ’;
        pb.kcrn[4] = ’ ’;pb.kcrn[5] = ’ ’; pb.kcrn[6] = ’ ’; pb.kcrn[7] =
’ ’;
        pb.kcdf = 0;
        pb.kclm = size;
        KDCS(&pb, &data);

        pb.kcop[0] = ’P’;
        pb.kcop[1] = ’E’;
        pb.kcop[2] = ’N’;
        pb.kcop[3] = ’D’;
        pb.kcom[0] = ’F’;
        pb.kcom[1] = ’I’;
        KDCS(&pb);
}


// Transact ORDER_STATUS
void KORDER_STATUS(struct kc_ca *x_kb, struct work *x_spab)
{
        PECBINFO pECBInfo;
        int size;
        kb = x_kb;
    spab = x_spab;

    pb.kcop[0] = ’I’;
```

```
        pb.kcop[1] = ’N’;
        pb.kcop[2] = ’I’;
        pb.kcop[3] = ’T’;
        pb.kclcapa = 0;
        pb.kclspa  = sizeof(struct work);
        KDCS (&pb);

        // read data - length in KBRFLD.kcrlm
        pb.kcop[0] = ’M’;
        pb.kcop[1] = ’G’;
        pb.kcop[2] = ’E’;
        pb.kcop[3] = ’T’;
        pb.kcla = sizeof(data);
        pb.kcfn[0] = ’ ’;pb.kcfn[1] = ’ ’; pb.kcfn[2] = ’ ’; pb.kcfn[3] =
’ ’;
        pb.kcfn[4] = ’ ’;pb.kcfn[5] = ’ ’; pb.kcfn[6] = ’ ’; pb.kcfn[7] =
’ ’;
        KDCS( &pb, &data);

        pECBInfo = SQLGetECB(pdbproc);
        size = KBRFLD.kcrlm;

        Trace("Beginning ORDER_ STATUS transaction, kcrlm=%d\n", size);

        data.Error = 0;
        data.Return = SQLOrderStatus(NULL, data.TermId, data.SyncId,
pdbproc,

&data.Trans.OrderStatusData, data.DeadlockRetry);
        data.bDeadlock = pECBInfo->bDeadlock;
        data.bFailed = pECBInfo->bFailed;
        if (data.Error)
        {
                strcpy(data.Trans.ErrorMsg, ErrorMsgBuffer);
        }

        Trace("Finished ORDER_ STATUS transaction\n");

        pb.kcop[0] = ’M’;
        pb.kcop[1] = ’P’;
        pb.kcop[2] = ’U’;
        pb.kcop[3] = ’T’;
        pb.kcom[0] = ’N’;
        pb.kcom[1] = ’T’;
        pb.kcfn[0] = ’ ’;pb.kcfn[1] = ’ ’; pb.kcfn[2] = ’ ’; pb.kcfn[3] =
’ ’;
        pb.kcfn[4] = ’ ’;pb.kcfn[5] = ’ ’; pb.kcfn[6] = ’ ’; pb.kcfn[7] =
’ ’;
        pb.kcrn[0] = ’ ’;pb.kcrn[1] = ’ ’; pb.kcrn[2] = ’ ’; pb.kcrn[3] =
’ ’;
        pb.kcrn[4] = ’ ’;pb.kcrn[5] = ’ ’; pb.kcrn[6] = ’ ’; pb.kcrn[7] =
’ ’;
        pb.kcdf = 0;
        pb.kclm = size;
        KDCS(&pb, &data);

        pb.kcop[0] = ’P’;
        pb.kcop[1] = ’E’;
        pb.kcop[2] = ’N’;
        pb.kcop[3] = ’D’;
        pb.kcom[0] = ’F’;
        pb.kcom[1] = ’I’;
        KDCS(&pb);
}

# Microsoft Developer Studio Generated NMAKE File, Format Version 4.10
# ** DO NOT EDIT **

# TARGTYPE "Win32 (x86) Console Application" 0x0103

!IF "$(CFG)" == ""
CFG=utm_server - Win32 Debug
!MESSAGE No configuration specified.  Defaulting to utm_server - Win32
Debug.
!ENDIF

!IF "$(CFG)" != "utm_server - Win32 Release" && "$(CFG)" !=\
 "utm_server - Win32 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 /f "utm_server.mak" CFG="utm_server - Win32 Debug"
!MESSAGE
!MESSAGE Possible choices for configuration are:
!MESSAGE
!MESSAGE "utm_server - Win32 Release" (based on\
 "Win32 (x86) Console Application")
!MESSAGE "utm_server - Win32 Debug" (based on\
 "Win32 (x86) Console Application")
!MESSAGE
!ERROR An invalid configuration is specified.
!ENDIF

!IF "$(OS)" == "Windows_NT"
NULL=
!ELSE
NULL=nul
!ENDIF
#########################################################################
#######
# Begin Project
# PROP Target_Last_Scanned "utm_server - Win32 Debug"
CPP=cl.exe
RSC=rc.exe
```

```
!IF  "$(CFG)" == "utm_server - Win32 Release"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 0
# PROP BASE Output_Dir "Release"
# PROP BASE Intermediate_Dir "Release"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 0
# PROP Output_Dir "Release"
# PROP Intermediate_Dir "Release"
# PROP Target_Dir ""
OUTDIR=.\Release
INTDIR=.\Release

ALL : ".\utmwork.exe"

CLEAN :
        -@erase "$(INTDIR)\error.obj"
        -@erase "$(INTDIR)\rSERV1.obj"
        -@erase "$(INTDIR)\utm_serv.obj"
        -@erase ".\utmwork.exe"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE"
/YX /c
# ADD CPP /nologo /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE" /YX /c
CPP_PROJ=/nologo /ML /W3 /GX /O2 /D "WIN32" /D "NDEBUG" /D "_CONSOLE"\
 /Fp"$(INTDIR)/utm_server.pch" /YX /Fo"$(INTDIR)/" /c
CPP_OBJS=.\Release/
CPP_SBRS=.\.
# ADD BASE RSC /l 0x409 /d "NDEBUG"
# ADD RSC /l 0x409 /d "NDEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/utm_server.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /machine:I386
# ADD LINK32 libwork.lib libcmt.lib kernel32.lib user32.lib gdi32.lib
winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib
uuid.lib odbc32.lib odbccp32.lib ntwdblib.lib /nologo /subsystem:console
/machine:I386 /out:"utmwork.exe"
LINK32_FLAGS=libwork.lib libcmt.lib kernel32.lib user32.lib gdi32.lib\
 winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib
oleaut32.lib\
 uuid.lib odbc32.lib odbccp32.lib ntwdblib.lib /nologo
/subsystem:console\
 /incremental:no /pdb:"$(OUTDIR)/utmwork.pdb" /machine:I386
/out:"utmwork.exe"
LINK32_OBJS= \
        "$(INTDIR)\error.obj" \
        "$(INTDIR)\rSERV1.obj" \
        "$(INTDIR)\utm_serv.obj" \
        ".\mainutm.obj" \
        ".\MSGPRIV.OBJ"

".\utmwork.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ELSEIF  "$(CFG)" == "utm_server - Win32 Debug"

# PROP BASE Use_MFC 0
# PROP BASE Use_Debug_Libraries 1
# PROP BASE Output_Dir "Debug"
# PROP BASE Intermediate_Dir "Debug"
# PROP BASE Target_Dir ""
# PROP Use_MFC 0
# PROP Use_Debug_Libraries 1
# PROP Output_Dir "Debug"
# PROP Intermediate_Dir "Debug"
# PROP Target_Dir ""
OUTDIR=.\Debug
INTDIR=.\Debug

ALL : ".\utmwork.exe"

CLEAN :
        -@erase "$(INTDIR)\error.obj"
        -@erase "$(INTDIR)\rSERV1.obj"
        -@erase "$(INTDIR)\utm_serv.obj"
        -@erase "$(INTDIR)\vc40.idb"
        -@erase "$(INTDIR)\vc40.pdb"
        -@erase "$(OUTDIR)\utmwork.pdb"
        -@erase ".\utmwork.exe"
        -@erase ".\utmwork.ilk"

"$(OUTDIR)" :
    if not exist "$(OUTDIR)/$(NULL)" mkdir "$(OUTDIR)"

# ADD BASE CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_CONSOLE" /YX /c
# ADD CPP /nologo /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_CONSOLE" /YX /c
CPP_PROJ=/nologo /MLd /W3 /Gm /GX /Zi /Od /D "WIN32" /D "_DEBUG" /D
"_CONSOLE"\
 /Fp"$(INTDIR)/utm_server.pch" /YX /Fo"$(INTDIR)/" /Fd"$(INTDIR)/" /c
```

```
CPP_OBJS=.\Debug/
CPP_SBRS=.\.
# ADD BASE RSC /l 0x409 /d "_DEBUG"
# ADD RSC /l 0x409 /d "_DEBUG"
BSC32=bscmake.exe
# ADD BASE BSC32 /nologo
# ADD BSC32 /nologo
BSC32_FLAGS=/nologo /o"$(OUTDIR)/utm_server.bsc"
BSC32_SBRS= \

LINK32=link.exe
# ADD BASE LINK32 kernel32.lib user32.lib gdi32.lib winspool.lib
comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib uuid.lib
odbc32.lib odbccp32.lib /nologo /subsystem:console /debug /machine:I386
# ADD LINK32 libwork.lib libcmt.lib kernel32.lib user32.lib gdi32.lib
winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib oleaut32.lib
uuid.lib odbc32.lib odbccp32.lib ntwdblib.lib /nologo /subsystem:console
/debug /machine:I386 /out:"utmwork.exe"
LINK32_FLAGS=libwork.lib libcmt.lib kernel32.lib user32.lib gdi32.lib\
 winspool.lib comdlg32.lib advapi32.lib shell32.lib ole32.lib
oleaut32.lib\
 uuid.lib odbc32.lib odbccp32.lib ntwdblib.lib /nologo
/subsystem:console\
 /incremental:yes /pdb:"$(OUTDIR)/utmwork.pdb" /debug /machine:I386\
 /out:"utmwork.exe"
LINK32_OBJS= \
	"$(INTDIR)\error.obj" \
	"$(INTDIR)\rSERV1.obj" \
	"$(INTDIR)\utm_serv.obj" \
	".\mainutm.obj" \
	".\MSGPRIV.OBJ"

".\utmwork.exe" : "$(OUTDIR)" $(DEF_FILE) $(LINK32_OBJS)
    $(LINK32) @<<
  $(LINK32_FLAGS) $(LINK32_OBJS)
<<

!ENDIF

.c{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.cxx{$(CPP_OBJS)}.obj:
   $(CPP) $(CPP_PROJ) $<

.c{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

.cpp{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<
```

```
.cxx{$(CPP_SBRS)}.sbr:
   $(CPP) $(CPP_PROJ) $<

########################################################################
#######
# Begin Target

# Name "utm_server - Win32 Release"
# Name "utm_server - Win32 Debug"

!IF  "$(CFG)" == "utm_server - Win32 Release"

!ELSEIF  "$(CFG)" == "utm_server - Win32 Debug"

!ENDIF

########################################################################
#######
# Begin Source File

SOURCE=.\utm_serv.c

!IF  "$(CFG)" == "utm_server - Win32 Release"

DEP_CPP_UTM_S=\
	{$(INCLUDE)}"\kcapro.h"\
	{$(INCLUDE)}"\kcca.h"\
	{$(INCLUDE)}"\kcdf.h"\
	{$(INCLUDE)}"\kcmac.h"\
	{$(INCLUDE)}"\kcop.h"\
	{$(INCLUDE)}"\kcpa.h"\
	{$(INCLUDE)}"\SQLDB.H"\
	{$(INCLUDE)}"\SQLFRONT.H"\
	{$(INCLUDE)}"\sqlroutines.h"\
	{$(INCLUDE)}"\tpcc.h"\
	{$(INCLUDE)}"\tpcc_org.h"\
	{$(INCLUDE)}"\trans.h"\
	{$(INCLUDE)}"\utm.h"\
	{$(INCLUDE)}"\XATMI.H"\
	{$(INCLUDE)}"\XATMIDEF.H"\


"$(INTDIR)\utm_serv.obj" : $(SOURCE) $(DEP_CPP_UTM_S) "$(INTDIR)"


!ELSEIF  "$(CFG)" == "utm_server - Win32 Debug"

DEP_CPP_UTM_S=\
	{$(INCLUDE)}"\kcapro.h"\
	{$(INCLUDE)}"\kcca.h"\
	{$(INCLUDE)}"\kcdf.h"\
	{$(INCLUDE)}"\kcmac.h"\
        {$(INCLUDE)}"\kcop.h"\
        {$(INCLUDE)}"\kcpa.h"\
        {$(INCLUDE)}"\SQLDB.H"\
        {$(INCLUDE)}"\SQLFRONT.H"\
        {$(INCLUDE)}"\sqlroutines.h"\
        {$(INCLUDE)}"\tpcc.h"\
        {$(INCLUDE)}"\tpcc_org.h"\
        {$(INCLUDE)}"\trans.h"\
        {$(INCLUDE)}"\utm.h"\
        {$(INCLUDE)}"\XATMI.H"\
        {$(INCLUDE)}"\XATMIDEF.H"\


"$(INTDIR)\utm_serv.obj" : $(SOURCE) $(DEP_CPP_UTM_S) "$(INTDIR)"


!ENDIF

# End Source File
#########################################################################
#######
# Begin Source File

SOURCE=.\rSERV1.c
DEP_CPP_RSERV=\
        {$(INCLUDE)}"\kcca.h"\
        {$(INCLUDE)}"\kccf.h"\
        {$(INCLUDE)}"\kcinp.h"\
        {$(INCLUDE)}"\kcpa.h"\
        {$(INCLUDE)}"\kctypdef.h"\
        {$(INCLUDE)}"\xiipc.h"\
        {$(INCLUDE)}"\xiiutmdb.h"\
        {$(INCLUDE)}"\xiiutmfo.h"\
        {$(INCLUDE)}"\xiiutmhl.h"\
        {$(INCLUDE)}"\xiletter.h"\
        {$(INCLUDE)}"\xirtcc.h"\
        {$(INCLUDE)}"\xirtcprt.h"\
        {$(INCLUDE)}"\xirtdata.h"\
        {$(INCLUDE)}"\xirtdev.h"\
        {$(INCLUDE)}"\xirtend.h"\
        {$(INCLUDE)}"\xirtstrt.h"\
        {$(INCLUDE)}"\xitam.h"\
        {$(INCLUDE)}"\xitskm.h"\


"$(INTDIR)\rSERV1.obj" : $(SOURCE) $(DEP_CPP_RSERV) "$(INTDIR)"


# End Source File
#########################################################################
#######
# Begin Source File
```

```
SOURCE=.\MSGPRIV.OBJ

!IF  "$(CFG)" == "utm_server - Win32 Release"

!ELSEIF  "$(CFG)" == "utm_server - Win32 Debug"

!ENDIF

# End Source File
#########################################################################
#######
# Begin Source File

SOURCE=.\mainutm.obj

!IF  "$(CFG)" == "utm_server - Win32 Release"

!ELSEIF  "$(CFG)" == "utm_server - Win32 Debug"

!ENDIF

# End Source File
#########################################################################
#######
# Begin Source File

SOURCE="\openUTM-SRC\AUDIT\shared\error.c"
DEP_CPP_ERROR=\
        {$(INCLUDE)}"\SQLDB.H"\
        {$(INCLUDE)}"\SQLFRONT.H"\
        {$(INCLUDE)}"\tpcc.h"\
        {$(INCLUDE)}"\tpcc_org.h"\
        {$(INCLUDE)}"\trans.h"\
        {$(INCLUDE)}"\util.h"\


"$(INTDIR)\error.obj" : $(SOURCE) $(DEP_CPP_ERROR) "$(INTDIR)"
   $(CPP) $(CPP_PROJ) $(SOURCE)


# End Source File
# End Target
# End Project
#########################################################################
#######
```

Appendix B - Database Details

```
/*  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 tpcsc1=3530,
          tpcsc2=3530,
          tpcsc3=3530,
          tpcsc4=3530,
          tpcsc5=3530,

          tpcsc1=3530,
          tpcsc2=3530,
          tpcsc3=3530,
          tpcsc4=3530,
          tpcsc5=3530,

          tpcsc1=3530,
          tpcsc2=3530,
          tpcsc3=3530,
          tpcsc4=3530,
          tpcsc5=3530,

          tpcol1=3000,
          tpcol2=3000,
          tpcol3=3000,
          tpcol4=3000,

          tpcol1=3000,
          tpcol2=3000,
          tpcol3=3000,
          tpcol4=3000,

          tpcmisc5=1000,
          tpcmisc4=1000,
          tpcmisc3=1000,
          tpcmisc2=1000,
          tpcmisc1=1000

          log on tpclog1=22000
go
```

```
/*  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
```

```
/*  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

/*  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

            /* accummulate lineitem amounts for this order into customer
*/

                   update customer
                      set c_balance      = c_balance + @total,
                          c_delivery_cnt = c_delivery_cnt + 1

            where c_w_id = @w_id and
                         c_d_id = @d_id and
                             c_id   = @c_id

        end

        select @oid1 = case @d_id when  1  then @o_id else @oid1 end,
               @oid2 = case @d_id when  2  then @o_id else @oid2 end,
               @oid3 = case @d_id when  3  then @o_id else @oid3 end,
               @oid4 = case @d_id when  4  then @o_id else @oid4 end,
               @oid5 = case @d_id when  5  then @o_id else @oid5 end,
               @oid6 = case @d_id when  6  then @o_id else @oid6 end,
               @oid7 = case @d_id when  7  then @o_id else @oid7 end,
               @oid8 = case @d_id when  8  then @o_id else @oid8 end,
               @oid9 = case @d_id when  9  then @o_id else @oid9 end,
               @oid10 = case @d_id when 10 then @o_id else @oid10 end

    end

commit tran d

select @oid1,
       @oid2,
       @oid3,
       @oid4,
       @oid5,
       @oid6,
       @oid7,
       @oid8,
       @oid9,
       @oid10

go

/*  TPC-C Benchmark Kit                                */
```

```
/*                                                  */
/*  DISKINIT.SQL                                    */
/*                                                  */
/*  This script is used create devices              */


use master
go

/* device for log 22,000 MB */
disk init name = "tpclog1",
       physname   = "l:",
       vdevno     = 14,
       size       = 11264000
go


/* device for Customer and stock */
disk init name = "tpcsc1",
       physname   = "g:",
       vdevno     = 15,
       size       = 5427200
go

disk init name = "tpcsc2",
       physname   = "i:",
       vdevno     = 16,
       size       = 5427200
go

disk init name = "tpcsc3",
       physname   = "k:",
       vdevno     = 17,
       size       = 5427200
go

disk init name = "tpcsc4",
       physname   = "n:",
       vdevno     = 18,
       size       = 5427200
go

disk init name = "tpcsc5",
       physname   = "p:",
       vdevno     = 19,
       size       = 5427200
go

/*  Devices for warehouse, district, item, new order, history, orders */
disk init name = "tpcmisc1",
       physname   = "f:",
       vdevno     = 20,
       size       = 512000
```

```
go

disk init name = "tpcmisc2",
       physname   = "h:",
       vdevno     = 21,
       size       = 512000
go

disk init name = "tpcmisc3",
       physname   = "j:",
       vdevno     = 22,
       size       = 512000
go

disk init name = "tpcmisc4",
       physname   = "m:",
       vdevno     = 23,
       size       = 512000
go

disk init name = "tpcmisc5",
       physname   = "o:",
       vdevno     = 24,
       size       = 512000
go

/* device for order line */
disk init name = "tpcol1",
       physname = "q:",
       vdevno   = 25,
       size     = 3072000
go

disk init name = "tpcol2",
       physname = "r:",
       vdevno   = 26,
       size     = 3072000
go

disk init name = "tpcol3",
       physname = "s:",
       vdevno   = 27,
       size     = 3072000
go

disk init name = "tpcol4",
       physname = "t:",
       vdevno   = 28,
       size     = 3072000
go
```

```
/*  File:       NEWORD.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 availible 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
                       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,"",0,0
                       select @commit_flag = 0

               end
       end
```

```
    /* get customer last name, discount, and credit rating */

    select @c_last     = c_last,
           @c_discount = c_discount,
           @c_credit   = c_credit,
                  @c_id_local = c_id
    from customer 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:       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


/*  File:       PAYMENT.SQL
*/
/*              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_balance - @h_amount,
                c_payment_cnt  = c_payment_cnt + 1,
                c_ytd_payment  = c_ytd_payment + @h_amount,
                @c_first       = c_first,
                @c_middle      = c_middle,
            @c_last        = c_last,
            @c_street_1    = c_street_1,
                @c_street_2    = c_street_2,
                @c_city        = c_city,
                @c_state       = c_state,
                @c_zip         = c_zip,
                @c_phone       = c_phone,
                @c_credit      = c_credit,
                @c_credit_lim  = c_credit_lim,
                @c_discount    = c_discount,
                @c_since       = c_since,
                @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
        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


/*  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

/*  TPC-C Benchmark Kit                                */
/*                                                     */
/*  SEGMENT.SQL                                        */
/*                                                     */
/*  This script is used create segments               */

use tpcc
go

/* create segment for warehouse, district, item tables, new order, history
orders */
sp_addsegment    misc_seg, tpcmisc1
go
sp_extendsegment    misc_seg, tpcmisc2
go
sp_extendsegment    misc_seg, tpcmisc3
go
sp_extendsegment    misc_seg, tpcmisc4
go
sp_extendsegment    misc_seg, tpcmisc5
go
sp_extendsegment    misc_seg, tpcmisc6
go

/* create segment for order-line table */
sp_addsegment    ol_seg, tpcol1
go
sp_extendsegment    ol_seg, tpcol2
go
sp_extendsegment    ol_seg, tpcol3
go
sp_extendsegment    ol_seg, tpcol4
go
sp_extendsegment    ol_seg, tpcol5
go
sp_extendsegment    ol_seg, tpcol6
go

/* create segment for customer & scock table */
sp_addsegment    sc_seg, tpcsc1
go
sp_extendsegment    sc_seg, tpcsc2
go
sp_extendsegment    sc_seg, tpcsc3
go
sp_extendsegment    sc_seg, tpcsc4
go
sp_extendsegment    sc_seg, tpcsc5
go
sp_extendsegment    sc_seg, tpcsc6
go

/*  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,
                                             @threshhold    smallint
as
declare @o_id_low int,
        @o_id_high int

    select @o_id_low  = (d_next_o_id - 20),
              @o_id_high = (d_next_o_id - 1)
    from district
    where d_w_id = @w_id and
                 d_id   = @d_id

    select count(distinct(s_i_id))
        from  stock, order_line
    where ol_w_id    =  @w_id and
             ol_d_id    =  @d_id and
             ol_o_id between @o_id_low and @o_id_high and
          s_w_id     =  ol_w_id and
             s_i_id     =  ol_i_id and
             s_quantity <  @threshhold

go



/*  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



/*  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



/*  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 sc_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 sc_seg
go
select getdate()
go




/*  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




/*  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

/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXNODCL.SQL
*/
/*
*/
/*  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



/*  TPC-C Benchmark Kit
*/
/*
*/
/*  IDXODLCL.SQL
*/
/*
*/
/*  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




/*  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





/*  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 sc_seg
```

```
go
select getdate()
go


/*  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




/*  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                            smallint,
       w_name                          char(10),
       w_street_1                      char(20),
       w_street_2                      char(20),
       w_city                          char(20),
       w_state                         char(2),
       w_zip                           char(9),
       w_tax                           numeric(4,4),
       w_ytd                           numeric(12,2)
) on 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                          smallint,
       d_name                          char(10),
       d_street_1                      char(20),
       d_street_2                      char(20),
       d_city                          char(20),
       d_state                         char(2),
       d_zip                           char(9),
       d_tax                           numeric(4,4),
       d_ytd                           numeric(12,2),
       d_next_o_id                     int
) on 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 sc_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                          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                         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                          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


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 sc_seg
go
```

Appendix C - Tunable Parameters and Options

This section discloses the Windows NT 4.0 Enterprise Edition registry parameters used on the Primergy 560 server systems.

```
HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer

Key Name:          SOFTWARE\Microsoft\MSSQLServer
Class Name:        <NO CLASS>
Last Write Time:   10/28/97 - 3:28 PM

Key Name:          SOFTWARE\Microsoft\MSSQLServer\Client
Class Name:        <NO CLASS>
Last Write Time:   10/28/97 - 3:40 PM

Key Name:          SOFTWARE\Microsoft\MSSQLServer\Client\ConnectTo
Class Name:        <NO CLASS>
Last Write Time:   11/10/97 - 8:54 PM
Value 0
  Name:            DSQUERY
  Type:            REG_SZ
  Data:            DBMSSOCN


Key Name:          SOFTWARE\Microsoft\MSSQLServer\Client\DB-Lib
Class Name:        <NO CLASS>
Last Write Time:   11/18/97 - 7:57 AM
Value 0
  Name:            AutoAnsiToOem
  Type:            REG_SZ
  Data:            on

Value 1
  Name:            UseIntlSettings
  Type:            REG_SZ
  Data:            ON


Key Name:          SOFTWARE\Microsoft\MSSQLServer\MSSQLServer
Class Name:        <NO CLASS>
Last Write Time:   11/14/97 - 8:55 AM
Value 0
  Name:            AuditLevel
  Type:            REG_DWORD
  Data:            0

Value 1
  Name:            DefaultDomain
  Type:            REG_SZ
  Data:            GEMINI

Value 2
  Name:            DefaultLogin
  Type:            REG_SZ
  Data:            guest

Value 3
  Name:            ImpersonateClient
  Type:            REG_DWORD
  Data:            0

Value 4
  Name:            ListenOn
  Type:            REG_MULTI_SZ
  Data:            SSMSSO60,1433


Value 5
  Name:            LoginMode
  Type:            REG_DWORD
  Data:            0

Value 6
  Name:            MailAccountName
  Type:            REG_SZ
  Data:

Value 7
  Name:            MailPassword
  Type:            REG_SZ
  Data:

Value 8
  Name:            Map#
  Type:            REG_SZ
  Data:            -

Value 9
  Name:            Map$
  Type:            REG_SZ
  Data:

Value 10
```

```
  Name:             Map_
  Type:             REG_SZ
  Data:             \

Value 11
  Name:             ResourceMgrID
  Type:             REG_SZ
  Data:             {B55413DC-4FA2-11D1-8979-0020AFED8EED}

Value 12
  Name:             SetHostname
  Type:             REG_DWORD
  Data:             0

Value 13
  Name:             Tapeloadwaittime
  Type:             REG_DWORD
  Data:             0xffffffff


Key Name:
SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\CurrentVersion
Class Name:         <NO CLASS>
Last Write Time:    10/29/97 - 12:40 PM
Value 0
  Name:             CurrentVersion
  Type:             REG_SZ
  Data:             6.50.258

Value 1
  Name:             RegisteredOrganization
  Type:             REG_SZ
  Data:             SNI

Value 2
  Name:             RegisteredOwner
  Type:             REG_SZ
  Data:             J. Schwarzmann

Value 3
  Name:             RegisteredProductID
  Type:             REG_SZ
  Data:

Value 4
  Name:             SerialNumber
  Type:             REG_DWORD
  Data:             0x85230040

Value 5
  Name:             SoftwareType
  Type:             REG_SZ
  Data:             System
```

```

Key Name:           SOFTWARE\Microsoft\MSSQLServer\MSSQLServer\Parameters
Class Name:         <NO CLASS>
Last Write Time:    11/14/97 - 8:55 AM
Value 0
  Name:             SQLArg0
  Type:             REG_SZ
  Data:             -dD:\MSSQL\DATA\MASTER.DAT

Value 1
  Name:             SQLArg1
  Type:             REG_SZ
  Data:             -eD:\MSSQL\LOG\ERRORLOG


Key Name:           SOFTWARE\Microsoft\MSSQLServer\Replication
Class Name:         <NO CLASS>
Last Write Time:    10/28/97 - 3:54 PM
Value 0
  Name:             DistributionDB
  Type:             REG_SZ
  Data:

Value 1
  Name:             WorkingDirectory
  Type:             REG_SZ
  Data:             D:\MSSQL\REPLDATA


Key Name:           SOFTWARE\Microsoft\MSSQLServer\Setup
Class Name:         <NO CLASS>
Last Write Time:    11/14/97 - 8:55 AM
Value 0
  Name:             CRC
  Type:             REG_SZ
  Data:             130877980

Value 1
  Name:             SetupStatus
  Type:             REG_SZ
  Data:             Installed

Value 2
  Name:             SQLPath
  Type:             REG_SZ
  Data:             D:\MSSQL


Key Name:           SOFTWARE\Microsoft\MSSQLServer\SQL Interface
Class Name:         REG_MULTI_SZ
Last Write Time:    11/4/97 - 1:42 PM
```

```
Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQL Interface\Graph
Control
Class Name:        REG_MULTI_SZ
Last Write Time:   11/4/97 - 1:42 PM

Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQL Service Manager
Class Name:        <NO CLASS>
Last Write Time:   10/29/97 - 4:14 PM
Value 0
  Name:            Action Verify
  Type:            REG_DWORD
  Data:            0

Value 1
  Name:            Background Interval
  Type:            REG_DWORD
  Data:            0x5

Value 2
  Name:            DefaultSvc
  Type:            REG_SZ
  Data:            MSSQLServer

Value 3
  Name:            Foreground Interval
  Type:            REG_DWORD
  Data:            0x2

Value 4
  Name:            Remote
  Type:            REG_DWORD
  Data:            0x1

Value 5
  Name:            Services
  Type:            REG_MULTI_SZ
  Data:            MSSQLServer
                   SQLExecutive
                   MSDTC


Value 6
  Name:            WindowDimensions
  Type:            REG_SZ
  Data:            0,262,193,275,214


Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQLExecutive
Class Name:        <NO CLASS>
Last Write Time:   10/28/97 - 3:54 PM
Value 0
  Name:            CmdExecAccount
  Type:            REG_BINARY
```

```
  Data:
00000000   5d 8d 63 92 3a 5e e2 25 - e1 39 99 64 91 d2 ef f7
].c.:^.%.9.d....

Value 1
  Name:            MailAutoStart
  Type:            REG_DWORD
  Data:            0x1

Value 2
  Name:            NonAlertableErrors
  Type:            REG_SZ
  Data:            1204,4002

Value 3
  Name:            RestartSQLServer
  Type:            REG_DWORD
  Data:            0x1

Value 4
  Name:            RestartSQLServerInterval
  Type:            REG_DWORD
  Data:            0x5

Value 5
  Name:            ServerHost
  Type:            REG_SZ
  Data:

Value 6
  Name:            SyshistoryLimitRows
  Type:            REG_DWORD
  Data:            0x1

Value 7
  Name:            SyshistoryMaxRows
  Type:            REG_DWORD
  Data:            0x3e8

Value 8
  Name:            TaskHistoryMaxRows
  Type:            REG_DWORD
  Data:            0x64


Key Name:          SOFTWARE\Microsoft\MSSQLServer\SQLExecutive\Subsystems
Class Name:        <NO CLASS>
Last Write Time:   10/28/97 - 3:54 PM
Value 0
  Name:            CmdExec
  Type:            REG_SZ
  Data:
D:\MSSQL\BINN\CMDEXEC.DLL,CmdExecStart,CmdEvent,CmdExecStop,10
```

```
Value 1
  Name:            Distribution
  Type:            REG_SZ
  Data:
D:\MSSQL\BINN\SQLREPL.DLL,distribution_start,distribution_event,distribut
ion_stop,100

Value 2
  Name:            LogReader
  Type:            REG_SZ
  Data:
D:\MSSQL\BINN\SQLREPL.DLL,logreader_start,logreader_event,logreader_stop,
25

Value 3
  Name:            Sync
  Type:            REG_SZ
  Data:
D:\MSSQL\BINN\SQLREPL.DLL,sync_start,sync_event,sync_stop,100



HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager

Key Name:          SYSTEM\CurrentControlSet\Control\Session Manager
Class Name:        <NO CLASS>
Last Write Time:   10/28/97 - 2:46 PM
Value 0
  Name:            BootExecute
  Type:            REG_MULTI_SZ
  Data:            autocheck autochk *


Value 1
  Name:            CriticalSectionTimeout
  Type:            REG_DWORD
  Data:            0x278d00

Value 2
  Name:            EnableMCA
  Type:            REG_DWORD
  Data:            0x1

Value 3
  Name:            EnableMCE
  Type:            REG_DWORD
  Data:            0

Value 4
  Name:            ExcludeFromKnownDlls
  Type:            REG_MULTI_SZ
  Data:
```

```
Value 5
  Name:            GlobalFlag
  Type:            REG_DWORD
  Data:            0

Value 6
  Name:            HeapDeCommitFreeBlockThreshold
  Type:            REG_DWORD
  Data:            0

Value 7
  Name:            HeapDeCommitTotalFreeThreshold
  Type:            REG_DWORD
  Data:            0

Value 8
  Name:            HeapSegmentCommit
  Type:            REG_DWORD
  Data:            0

Value 9
  Name:            HeapSegmentReserve
  Type:            REG_DWORD
  Data:            0

Value 10
  Name:            LicensedProcessors
  Type:            REG_DWORD
  Data:            0x4

Value 11
  Name:            ObjectDirectories
  Type:            REG_MULTI_SZ
  Data:            \Windows
                   \RPC Control


Value 12
  Name:            ProcessorControl
  Type:            REG_DWORD
  Data:            0x2

Value 13
  Name:            ProtectionMode
  Type:            REG_DWORD
  Data:            0

Value 14
  Name:            RegisteredProcessors
  Type:            REG_DWORD
  Data:            0x4
```

```
Value 15
  Name:            ResourceTimeoutCount
  Type:            REG_DWORD
  Data:            0x9e340


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\CWD
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\CWD\ff060102423da0000407108e0500
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\CWD\ff060102423da0000407108e0500\1
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            Add1
  Type:            REG_BINARY
  Data:
00000000   02 15 40 a0 10 1e b8 23 - 00 8e d8 8b 0e 14 07 81
..@....#........
00000010   e1 00 02 1f c3                                      .....

Value 1
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 1d 50 48 0c 55 8b ec - b8 00 00 9c 59 81 e1 00
..PH.U......Y...
00000010   02 55 8b ec b8 00 00 e8 - e7 57 90 90 90
.U.......W...


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\MYST
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\MYST\ff060102423bab000407102e0600
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
```

```
Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\MYST\ff060102423bab000407102e0600\1
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            Add1
  Type:            REG_BINARY
  Data:
00000000   02 15 40 ab 10 1e b8 23 - 00 8e d8 8b 0e 14 07 81
..@....#........
00000010   e1 00 02 1f c3                                      .....

Value 1
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 1d 50 49 0c 55 8b ec - b8 00 00 9c 59 81 e1 00
..PI.U......Y...
00000010   02 55 8b ec b8 00 00 e8 - e7 61 90 90 90
.U.......a...


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\PALED40
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\PALED40\ff06010242003200040740 1b0100
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\PALED40\ff0601024200320004074 01b0100\1
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 07 b7 21 01 d8 0c                                ...!...


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\USA
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\USA\ff06010242059b00040710780600
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
```

```
Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\USA\ff06010242059b00040710780600\1
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 1d 95 44 0c 55 8b ec - b8 00 00 9c 59 81 e1 00
...D.U......Y...
00000010   02 55 8b ec b8 00 00 e8 - 67 56 90 90 90
.U......gV...

Value 1
  Name:            Change2
  Type:            REG_BINARY
  Data:
00000000   01 25 05 9b 10 00 00 00 - 00 00 00 00 00 00 00 00
.%..............
00000010   00 00 00 00 00 1e b8 23 - 00 8e d8 8b 0e 14 07 81
.......#........
00000020   e1 00 02 1f c3                                   .....


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB\ff060102ec353f00040780c81300
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB\ff060102ec353f00040780c81300\12
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 11 1b 03 06 81 3e ba - 31 34 03 81 3e ba 31 09
......>.14..>.1.
00000010   03                                               .


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB40016
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
```

```
Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB40016\ff0702021401ee3e000407d0460e00
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\AppPatches\VB40016\ff0702021401ee3e000407d0460e00\16
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            Change1
  Type:            REG_BINARY
  Data:
00000000   01 11 6d 2a 06 81 3e 6e - 36 34 03 81 3e 6e 36 09
..m*..>n64..>n6.
00000010   03                                               .


Key Name:          SYSTEM\CurrentControlSet\Control\Session Manager\DOS
Devices
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            AUX
  Type:            REG_SZ
  Data:            \DosDevices\COM1

Value 1
  Name:            MAILSLOT
  Type:            REG_SZ
  Data:            \Device\MailSlot

Value 2
  Name:            NUL
  Type:            REG_SZ
  Data:            \Device\Null

Value 3
  Name:            PIPE
  Type:            REG_SZ
  Data:            \Device\NamedPipe

Value 4
  Name:            PRN
  Type:            REG_SZ
  Data:            \DosDevices\LPT1

Value 5
  Name:            UNC
  Type:            REG_SZ
  Data:            \Device\Mup
```

```
Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\Environment
Class Name:        <NO CLASS>
Last Write Time:   11/18/97 - 10:44 AM
Value 0
  Name:            ComSpec
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\cmd.exe

Value 1
  Name:            NUMBER_OF_PROCESSORS
  Type:            REG_SZ
  Data:            4

Value 2
  Name:            OS
  Type:            REG_SZ
  Data:            Windows_NT

Value 3
  Name:            Os2LibPath
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\os2\dll;

Value 4
  Name:            Path
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32;%SystemRoot%;;D:\MSSQL\BINN

Value 5
  Name:            PROCESSOR_ARCHITECTURE
  Type:            REG_SZ
  Data:            x86

Value 6
  Name:            PROCESSOR_IDENTIFIER
  Type:            REG_SZ
  Data:            x86 Family 6 Model 1 Stepping 9, GenuineIntel

Value 7
  Name:            PROCESSOR_LEVEL
  Type:            REG_SZ
  Data:            6

Value 8
  Name:            PROCESSOR_REVISION
  Type:            REG_SZ
  Data:            0109

Value 9
  Name:            windir
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%
```

```
Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\Executive
Class Name:        <NO CLASS>
Last Write Time:   11/18/97 - 9:41 AM
Value 0
  Name:            AdditionalCriticalWorkerThreads
  Type:            REG_DWORD
  Data:            0

Value 1
  Name:            AdditionalDelayedWorkerThreads
  Type:            REG_DWORD
  Data:            0

Value 2
  Name:            PriorityQuantumMatrix
  Type:            REG_BINARY
  Data:
00000000   52 d6 03 59 00 a3 02 00 - a6 e3 bc 01
R..Y........


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\FileRenameOperations
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM

Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\KnownDLLs
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            advapi32
  Type:            REG_SZ
  Data:            advapi32.dll

Value 1
  Name:            comdlg32
  Type:            REG_SZ
  Data:            comdlg32.dll

Value 2
  Name:            crtdll
  Type:            REG_SZ
  Data:            crtdll.dll

Value 3
  Name:            DllDirectory
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32
```

```
Value 4
  Name:             gdi32
  Type:             REG_SZ
  Data:             gdi32.dll

Value 5
  Name:             kernel32
  Type:             REG_SZ
  Data:             kernel32.dll

Value 6
  Name:             lz32
  Type:             REG_SZ
  Data:             lz32.dll

Value 7
  Name:             ole32
  Type:             REG_SZ
  Data:             ole32.dll

Value 8
  Name:             oleaut32
  Type:             REG_SZ
  Data:             oleaut32.dll

Value 9
  Name:             olecli32
  Type:             REG_SZ
  Data:             olecli32.dll

Value 10
  Name:             olecnv32
  Type:             REG_SZ
  Data:             olecnv32.dll

Value 11
  Name:             olesvr32
  Type:             REG_SZ
  Data:             olesvr32.dll

Value 12
  Name:             olethk32
  Type:             REG_SZ
  Data:             olethk32.dll

Value 13
  Name:             rpcrt4
  Type:             REG_SZ
  Data:             rpcrt4.dll

Value 14
  Name:             shell32
  Type:             REG_SZ
  Data:             shell32.dll

Value 15
  Name:             user32
  Type:             REG_SZ
  Data:             user32.dll

Value 16
  Name:             version
  Type:             REG_SZ
  Data:             version.dll


Key Name:           SYSTEM\CurrentControlSet\Control\Session
Manager\Memory Management
Class Name:         <NO CLASS>
Last Write Time:    10/28/97 - 3:55 PM
Value 0
  Name:             ClearPageFileAtShutdown
  Type:             REG_DWORD
  Data:             0

Value 1
  Name:             DisablePagingExecutive
  Type:             REG_DWORD
  Data:             0

Value 2
  Name:             IoPageLockLimit
  Type:             REG_DWORD
  Data:             0

Value 3
  Name:             LargeSystemCache
  Type:             REG_DWORD
  Data:             0

Value 4
  Name:             NonPagedPoolQuota
  Type:             REG_DWORD
  Data:             0

Value 5
  Name:             NonPagedPoolSize
  Type:             REG_DWORD
  Data:             0

Value 6
  Name:             PagedPoolQuota
  Type:             REG_DWORD
  Data:             0

Value 7
```

```
  Name:            PagedPoolSize
  Type:            REG_DWORD
  Data:            0

Value 8
  Name:            PagingFiles
  Type:            REG_MULTI_SZ
  Data:            D:\pagefile.sys 267


Value 9
  Name:            SecondLevelDataCache
  Type:            REG_DWORD
  Data:            0

Value 10
  Name:            SystemPages
  Type:            REG_DWORD
  Data:            0


Key Name:          SYSTEM\CurrentControlSet\Control\Session
Manager\SubSystems
Class Name:        <NO CLASS>
Last Write Time:   10/10/96 - 9:09 AM
Value 0
  Name:            Debug
  Type:            REG_EXPAND_SZ
  Data:

Value 1
  Name:            Kmode
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\win32k.sys

Value 2
  Name:            Optional
  Type:            REG_MULTI_SZ
  Data:            Os2
                   Posix


Value 3
  Name:            Os2
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\os2ss.exe

Value 4
  Name:            Posix
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\psxss.exe

Value 5
  Name:            Required
  Type:            REG_MULTI_SZ
  Data:            Debug
                   Windows


Value 6
  Name:            Windows
  Type:            REG_EXPAND_SZ
  Data:            %SystemRoot%\system32\csrss.exe
ObjectDirectory=\Windows SharedSection=1024,3072 Windows=On
SubSystemType=Windows ServerDll=basesrv,1
ServerDll=winsrv:UserServerDllInitialization,3
ServerDll=winsrv:ConServerDllInitialization,2 ProfileControl=Off
MaxRequestThreads=16
```

This section discloses the Windows NT 4.0 Enterprise Edition registry parameters used on the Primergy 160 client systems.

```
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\InetInfo
    Parameters
        BandwidthLevel = REG_DWORD 0xffffffff
        ListenBackLog = REG_DWORD 0x00000708
        PoolThreadsLimit = REG_DWORD 0x000001fe
        ThreadTimeout = REG_DWORD 0x00015180
        MaxPoolThreads = REG_DWORD 0x000001fe
        Filter
            FilterType = REG_DWORD 0x00000000
            NumGrantSites = REG_DWORD 0x00000000
            NumDenySites = REG_DWORD 0x00000000
        MimeMap
            text/html,htm,,h =
            image/gif,gif,,g =
            image/jpeg,jpg,,: =
            text/plain,txt,,0 =
            text/html,html,,h =
            image/jpeg,jpeg,,: =
            image/jpeg,jpe,,: =
            image/bmp,bmp,,: =
            application/octet-stream,*,,5 =
            application/pdf,pdf,,5 =
            application/octet-stream,bin,,5 =
            application/oda,oda,,5 =
            application/zip,zip,,9 =
            application/rtf,rtf,,5 =
            application/postscript,ps,,5 =
        application/postscript,ai,,5 =
        application/postscript,eps,,5 =
        application/mac-binhex40,hqx,,4 =
        application/msword,doc,,5 =
        application/msword,dot,,5 =
        application/winhlp,hlp,,5 =
        video/mpeg,mpeg,,; =
        video/mpeg,mpg,,; =
        video/mpeg,mpe,,; =
        video/x-msvideo,avi,,< =
        video/quicktime,qt,,; =
        video/quicktime,mov,,; =
        video/x-sgi-movie,movie,,< =
        x-world/x-vrml,wrl,,5 =
        x-world/x-vrml,xaf,,5 =
        x-world/x-vrml,xof,,5 =
        x-world/x-vrml,flr,,5 =
        x-world/x-vrml,wrz,,5 =
        application/x-director,dcr,,5 =
        application/x-director,dir,,5 =
        application/x-director,dxr,,5 =
        image/cis-cod,cod,,5 =
        image/x-cmx,cmx,,5 =
        application/envoy,evy,,5 =
        application/x-msaccess,mdb,,5 =
        application/x-mscardfile,crd,,5 =
        application/x-msclip,clp,,5 =
        application/octet-stream,exe,,5 =
        application/x-msexcel,xla,,5 =
        application/x-msexcel,xlc,,5 =
        application/x-msexcel,xlm,,5 =
        application/x-msexcel,xls,,5 =
        application/x-msexcel,xlt,,5 =
        application/x-msexcel,xlw,,5 =
        application/x-msmediaview,m13,,5 =
        application/x-msmediaview,m14,,5 =
        application/x-msmoney,mny,,5 =
        application/x-mspowerpoint,ppt,,5 =
        application/x-msproject,mpp,,5 =
        application/x-mspublisher,pub,,5 =
        application/x-msterminal,trm,,5 =
        application/x-msworks,wks,,5 =
        application/x-mswrite,wri,,5 =
        application/x-msmetafile,wmf,,5 =
        application/x-csh,csh,,5 =
        application/x-dvi,dvi,,5 =
        application/x-hdf,hdf,,5 =
        application/x-latex,latex,,5 =
        application/x-netcdf,nc,,5 =
        application/x-netcdf,cdf,,5 =
        application/x-sh,sh,,5 =
        application/x-tcl,tcl,,5 =
        application/x-tex,tex,,5 =
        application/x-texinfo,texinfo,,5 =
        application/x-texinfo,texi,,5 =
        application/x-troff,t,,5 =
        application/x-troff,tr,,5 =
        application/x-troff,roff,,5 =
        application/x-troff-man,man,,5 =
        application/x-troff-me,me,,5 =
        application/x-troff-ms,ms,,5 =
        application/x-wais-source,src,,7 =
        application/x-bcpio,bcpio,,5 =
        application/x-cpio,cpio,,5 =
        application/x-gtar,gtar,,9 =
        application/x-shar,shar,,5 =
        application/x-sv4cpio,sv4cpio,,5 =
        application/x-sv4crc,sv4crc,,5 =
        application/x-tar,tar,,5 =
        application/x-ustar,ustar,,5 =
        audio/basic,au,,< =
        audio/basic,snd,,< =
        audio/x-aiff,aif,,< =
        audio/x-aiff,aiff,,< =
        audio/x-aiff,aifc,,< =
        audio/x-wav,wav,,< =
        audio/x-pn-realaudio,ram,,< =
        image/ief,ief,,: =
        image/tiff,tiff,,: =
        image/tiff,tif,,: =
        image/x-cmu-raster,ras,,: =
        image/x-portable-anymap,pnm,,: =
        image/x-portable-bitmap,pbm,,: =
        image/x-portable-graymap,pgm,,: =
        image/x-portable-pixmap,ppm,,: =
        image/x-rgb,rgb,,: =
        image/x-xbitmap,xbm,,: =
        image/x-xpixmap,xpm,,: =
        image/x-xwindowdump,xwd,,: =
        text/html,stm,,h =
        text/plain,bas,,0 =
        text/plain,c,,0 =
        text/plain,h,,0 =
        text/richtext,rtx,,0 =
        text/tab-separated-values,tsv,,0 =
        text/x-setext,etx,,0 =
        application/x-perfmon,pmc,,5 =
        application/x-perfmon,pma,,5 =
        application/x-perfmon,pmr,,5 =
        application/x-perfmon,pml,,5 =
        application/x-perfmon,pmw,,5 =
Performance
    Library = infoctrs.DLL
    Open = OpenINFOPerformanceData
    Close = CloseINFOPerformanceData
    Collect = CollectINFOPerformanceData
```

```
        Last Counter = REG_DWORD 0x00000756
        Last Help = REG_DWORD 0x00000757
        First Counter = REG_DWORD 0x00000738
        First Help = REG_DWORD 0x00000739

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\INetMgr
    InstalledBy = INetStp
    Parameters
        MajorVersion = REG_DWORD 0x00000002
        MinorVersion = REG_DWORD 0x00000000
        HelpLocation = iisadmin\htmldocs\inetdocs.htm
        x = REG_DWORD 0x00000000
        y = REG_DWORD 0x00000100
        dx = REG_DWORD 0x000001ac
        dy = REG_DWORD 0x000000b1
        Mode = REG_DWORD 0x00000001
        View = REG_DWORD 0x0000800b
        WaitTime = REG_DWORD 0x00007530
        AddOnServices
            FTP = fscfg.dll
            Gopher = gscfg.dll
            WWW = w3scfg.dll
        AddOnTools
            &Key Manager = C:\WINNT\System32\inetsrv\keyring.exe;Key
Manager

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Inetsrv
    CurrentVersion
        SoftwareType = service
        MajorVersion = REG_DWORD 0x00000004
        MinorVersion = REG_DWORD 0x00000000
        Title = Microsoft Internet Information Server 3.0
        Description = Microsoft Internet Information Server 3.0
        ServiceName = Microsoft Internet Information Server 3.0
        OperationsSupport = REG_DWORD 0x00000086
        InstallDate = REG_DWORD 0x33a041c2
        NetRules
            InfName = oemnsvin.inf
            InfOption = Inetsrv

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\MSSQLServer
    Client
        ConnectTo
            DSQUERY = DBMSSOCN
        DB-Lib
            AutoAnsiToOem = ON
            UseIntlSettings = ON
    ClientSetup
        SQLPath = C:\MSSQL

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager
    ObjectDirectories = REG_MULTI_SZ "\Windows"  \
                                     "\RPC Control"
```

```
    GlobalFlag = REG_DWORD 0x00000000
    ProtectionMode = REG_DWORD 0x00000000
    BootExecute = REG_MULTI_SZ "autocheck autochk *"
    EnableMCE = REG_DWORD 0x00000000
    EnableMCA = REG_DWORD 0x00000001
    HeapSegmentReserve = REG_DWORD 0x00000000
    HeapSegmentCommit = REG_DWORD 0x00000000
    HeapDeCommitTotalFreeThreshold = REG_DWORD 0x00000000
    HeapDeCommitFreeBlockThreshold = REG_DWORD 0x00000000
    CriticalSectionTimeout = REG_DWORD 0x00278d00
    ResourceTimeoutCount = REG_DWORD 0x0009e340
    ExcludeFromKnownDlls = REG_MULTI_SZ
    ProcessorControl = REG_DWORD 0x00000002
    RegisteredProcessors = REG_DWORD 0x00000004
    LicensedProcessors = REG_DWORD 0x00000004
    AppPatches
        CWD
            ff060102423da0000407108e0500
                1
                    Add1 = REG_BINARY 0x00000015 0xa0401502 0x23b81e10
0x8bd88e00 0x8107140e 0x1f0200e1 0x000000c3
                    Change1 = REG_BINARY 0x0000001d 0x48501d01 0xec8b550c
0x9c0000b8 0x00e18159 0xec8b5502 0xe80000b8 0x909057e7 0x00000090
        MYST
            ff060102423bab000407102e0600
                1
                    Add1 = REG_BINARY 0x00000015 0xab401502 0x23b81e10
0x8bd88e00 0x8107140e 0x1f0200e1 0x000000c3
                    Change1 = REG_BINARY 0x0000001d 0x49501d01 0xec8b550c
0x9c0000b8 0x00e18159 0xec8b5502 0xe80000b8 0x909061e7 0x00000090
        PALED40
            ff06010242003200040740 1b0100
                1
                    Change1 = REG_BINARY 0x00000007 0x21b70701 0x000cd801
        USA
            ff06010242059b00040710780600
                1
                    Change1 = REG_BINARY 0x0000001d 0x44951d01 0xec8b550c
0x9c0000b8 0x00e18159 0xec8b5502 0xe80000b8 0x90905667 0x00000090
                    Change2 = REG_BINARY 0x00000025 0x9b052501 0x00000010
0x00000000 0x00000000 0x00000000 0x23b81e00 0x8bd88e00 0x8107140e
0x1f0200e1 0x000000c3
        VB
            ff060102ec353f00040780c81300
                12
                    Change1 = REG_BINARY 0x00000011 0x031b1101 0xba3e8106
0x81033431 0x0931ba3e 0x00000003
        VB40016
            ff0702021401ee3e000407d0460e00
                16
                    Change1 = REG_BINARY 0x00000011 0x2a6d1101 0x6e3e8106
0x81033436 0x09366e3e 0x00000003
    DOS Devices
```

```
        PRN = \DosDevices\LPT1
        AUX = \DosDevices\COM1
        NUL = \Device\Null
        PIPE = \Device\NamedPipe
        MAILSLOT = \Device\MailSlot
        UNC = \Device\Mup
    Environment
        ComSpec = REG_EXPAND_SZ %SystemRoot%\system32\cmd.exe
        NUMBER_OF_PROCESSORS = 1
        NUT_DEFAULT_WIN32_FAULT = 1
        NUT_HEAP_RESERVE = 0
        NUT_SUFFIXED_SEARCHING = 0
        OS = Windows_NT
        Os2LibPath = REG_EXPAND_SZ %SystemRoot%\system32\os2\dll;
        Path = REG_EXPAND_SZ
%SystemRoot%\system32;%SystemRoot%;;C:\MSSQL\BINN;C:\PROGRA~1\COMMON~1\Sy
stem
        PROCESSOR_ARCHITECTURE = x86
        PROCESSOR_IDENTIFIER = x86 Family 6 Model 1 Stepping 9,
GenuineIntel
        PROCESSOR_LEVEL = 6
        PROCESSOR_REVISION = 0109
        UTM_MAIN_KILL_TIME = REG_EXPAND_SZ 1
        UTM_NET_SELECT_TIME = REG_EXPAND_SZ 100
        UTM_OSS_SHM_BASE = REG_EXPAND_SZ 0x00000000
        UTMPATH = REG_EXPAND_SZ C:\openUTM-Server
        windir = REG_EXPAND_SZ %SystemRoot%
    Executive [8 1 15 12 17 5]
        AdditionalCriticalWorkerThreads = REG_DWORD 0x00000000
        AdditionalDelayedWorkerThreads = REG_DWORD 0x00000000
        PriorityQuantumMatrix = REG_BINARY 0x0000000c 0x778c3e50
0x00000000 0x01bc7714
    FileRenameOperations
    KnownDLLs
        DllDirectory = REG_EXPAND_SZ %SystemRoot%\system32
        kernel32 = kernel32.dll
        gdi32 = gdi32.dll
        user32 = user32.dll
        rpcrt4 = rpcrt4.dll
        advapi32 = advapi32.dll
        comdlg32 = comdlg32.dll
        crtdll = crtdll.dll
        shell32 = shell32.dll
        lz32 = lz32.dll
        olecli32 = olecli32.dll
        olesvr32 = olesvr32.dll
        version = version.dll
        ole32 = ole32.dll
        oleaut32 = oleaut32.dll
        olecnv32 = olecnv32.dll
        olethk32 = olethk32.dll
        SHDOCVW = SHDOCVW.DLL
        SHLWAPI = SHLWAPI.DLL
```

```
    Memory Management [8 1 17]
        PagedPoolSize = REG_DWORD 0x00000000
        NonPagedPoolSize = REG_DWORD 0x00000000
        PagedPoolQuota = REG_DWORD 0x00000000
        NonPagedPoolQuota = REG_DWORD 0x00000000
        IoPageLockLimit = REG_DWORD 0x00000000
        LargeSystemCache = REG_DWORD 0x00000000
        PagingFiles = REG_MULTI_SZ "C:\pagefile.sys 128 178"  \
                                   "S:\pagefile.sys 1024 1024"
        SystemPages = REG_DWORD 0x00000000
        SecondLevelDataCache = REG_DWORD 0x00000000
        DisablePagingExecutive = REG_DWORD 0x00000000
        ClearPageFileAtShutdown = REG_DWORD 0x00000000
    SubSystems
        Required = REG_MULTI_SZ "Debug"  \
                                "Windows"
        Optional = REG_MULTI_SZ "Os2"  \
                                "Posix"
        Debug = REG_EXPAND_SZ
        Windows = REG_EXPAND_SZ %SystemRoot%\system32\csrss.exe
ObjectDirectory=\Windows SharedSection=1024,3072 Windows=On
SubSystemType=Windows ServerDll=basesrv,1
ServerDll=winsrv:UserServerDllInitialization,3 \

ServerDll=winsrv:ConServerDllInitialization,2 ProfileControl=Off \
                                MaxRequestThreads=16
        Kmode = REG_EXPAND_SZ %SystemRoot%\system32\win32k.sys
        Os2 = REG_EXPAND_SZ %SystemRoot%\system32\os2ss.exe
        Posix = REG_EXPAND_SZ %SystemRoot%\system32\psxss.exe

HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\TPCC
    PATH = C:\InetPub\wwwroot\
    LOG = OFF
    NumberOfDeliveryThreads = 3
    MaximumWarehouses = 900
    BackoffDelay = 500
    DeadlockRetry = 3
    MaxConnections = 1800
    QueueSlotts = 3000

HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\W3SVC [17 1]
    Type = REG_DWORD 0x00000020
    Start = REG_DWORD 0x00000003
    ErrorControl = REG_DWORD 0x00000000
    ImagePath = REG_EXPAND_SZ C:\WINNT\System32\inetsrv\inetinfo.exe
    DisplayName = World Wide Web Publishing Service
    DependOnService = REG_MULTI_SZ "RPCSS"  \
                                   "NTLMSSP"
    DependOnGroup = REG_MULTI_SZ
    ObjectName = LocalSystem
    Parameters
        MajorVersion = REG_DWORD 0x00000002
        MinorVersion = REG_DWORD 0x00000000
        AdminName = Administrator
        AdminEmail = Admin@corp.com
        MaxConnections = REG_DWORD 0x000186a0
        LogType = REG_DWORD 0x00000000
        LogFileDirectory = REG_EXPAND_SZ %SystemRoot%\System32\LogFiles
        LogFileTruncateSize = REG_DWORD 0x01388000
        LogFilePeriod = REG_DWORD 0x00000001
        LogFileFormat = REG_DWORD 0x00000000
        LogSqlDataSource = HTTPLOG
        LogSqlTableName = Internetlog
        LogSqlUserName = InternetAdmin
        LogSqlPassword = sqllog
        Authorization = REG_DWORD 0x00000003
        AnonymousUserName = IUSR_ORANGE
        Default Load File = Default.htm
        Dir Browse Control = REG_DWORD 0x4000001e
        CheckForWAISDB = REG_DWORD 0x00000000
        CacheExtensions = REG_DWORD 0x00000001
        GlobalExpire = REG_DWORD 0xffffffff
        ServerSideIncludesEnabled = REG_DWORD 0x00000001
        ServerSideIncludesExtension = .stm
        DebugFlags = REG_DWORD 0x00000008
        ScriptTimeout = REG_DWORD 0x00000384
        ConnectionTimeOut = REG_DWORD 0x00000384
        InstallPath = C:\WINNT\System32\inetsrv
        SecurePort = REG_DWORD 0x000001bb
        Filter DLLs = C:\WINNT\System32\inetsrv\sspifilt.dll
        AccessDeniedMessage = Error: Access is Denied.
        NTAuthenticationProviders = NTLM
        AcceptExOutstanding = REG_DWORD 0x00000708
        UsePoolThreadForCGI = REG_DWORD 0x00000001
        ServerComment =
        Script Map
            .idc = C:\WINNT\System32\inetsrv\httpodbc.dll
        Virtual Roots
            /, = C:\InetPub\wwwroot,,5
            /Scripts, = C:\InetPub\scripts,,4
            /iisadmin, = C:\WINNT\System32\inetsrv\iisadmin,,1
    Performance
        Library = w3ctrs.DLL
        Open = OpenW3PerformanceData
        Close = CloseW3PerformanceData
        Collect = CollectW3PerformanceData
        Last Counter = REG_DWORD 0x00000790
        Last Help = REG_DWORD 0x00000791
        First Counter = REG_DWORD 0x00000758
        First Help = REG_DWORD 0x00000759
    Security [17 1]
        Security = REG_BINARY 0x000000d8 0x80140001 0x000000c0 0x000000cc
0x00000014 0x00000034 0x00200002 0x00000001 0x00188002 0x000f01ff
0x00000101 0x01000000 0x00000000 0x00000220 0x008c0002 0x00000005
0x00180000 0x0002018d 0x00000101 \
                   0x01000000 0x00000000 0x0063006d 0x001c0000 0x000201fd
0x00000201 0x05000000 0x00000020 0x00000223 0x00610069 0x001c0000
0x000f01ff 0x00000201 0x05000000 0x00000020 0x00000220 0x00610069
0x001c0000 0x000f01ff 0x00000201 \
                   0x05000000 0x00000020 0x00000225 0x00610069 0x00180000
0x000201fd 0x00000101 0x05000000 0x00000012 0x00000225 0x00000101
0x05000000 0x00000012 0x00000101 0x05000000 0x00000012
    W3SAMP
    Enum
        0 = Root\LEGACY_W3SVC\0000
        Count = REG_DWORD 0x00000001
        NextInstance = REG_DWORD 0x00000001
```

This section discloses the Transaction monitor tunable parameters parameters used on the Primergy 160 client system.

```
.UTM START FILEBASE=.
.UTM START TASKS=27
.UTM START ASYNTASKS=40
.UTM START TESTMODE=OFF
.UTM START MULTI-PROC-OPT=OFF
.UTM END
```

This section discloses the Microsoft SQL Server 6.5 Enterprise Edition parameters used on the Primergy 560 server system.

After building the benchmark database we startet the following script to improve cache performance of the customer table and stock table:

```
use tpcc
go
update sysobjects set cache=2 from sysobjects where name = ‘stock’
go
update sysobjects set cache=5 from sysobjects where name = ‘customer’
go
```

Microsoft SQL Server was started with the following command line options:

sqlservr -c -x -t1081 –T1140 -T3502 -T812 -Cd1450000 -Cp5000

where:

-c starts SQL Server independently of the Windows NT Service Control Manager

-x disables the keeping of CPU time and cache-hit ratio statistics

-t1081 allows the index pages a "second" trip through the cache

-T812 omits sorting for write page ordering during checkpoints

-T1140 optimizes free space allocation

-T3502 prints a message to the SQL Server log at start and end of each checkpoint

-Cd1450000 specifies the number of data pages to allocate

-Cp5000 specifies the number of procedure cache pages to allocate

The following Microsoft SQL Server configuration parameters were used:

```
name                                minimum     maximum     config_value run_value
----------------------------------- ----------- ----------- ------------ -----------
affinity mask                                 0  2147483647           15          15
allow updates                                 0           1            1           1
backup buffer size                            1          32           10          10
backup threads                                0          32            0           0
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     1000000       930000      930011
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         1800        1800
logwrite sleep (ms)                          -1         500           -1          -1
max async IO                                  1        1024           32          32
max lazywrite IO                              1        1024           64          64
max text repl size                            0  2147483647        65536       65536
max worker threads                           10        1024          150         150
media retention                               0         365            0           0
memory                                     2800     1048576       950000      950000
nested triggers                               0           1            1           1
network packet size                         512       32767         1024        1024
open databases                                5       32767           10          10
open objects                                100  2147483647          450         450
```

```
priority boost                          0          1          0          0
procedure cache                         1         99          1          1
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          1          1
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        250        250
user options                            0       4095          0          0
```

This section additionally discloses hardware information of the Primergy 560 server system.

```
                              Board Information


SIEMENS NIXDORF  - PRIMERGY -  (D887)

          System

The PCD-6T System board provides:
- up to 4 CPU's
- up to 2GB RAM
- 2 PCI-buses (33MHz)
- floppy controller
- IDE controller
- Server Managment features
- two async. communication ports (V24)
- one parallel port (Centronics)

Manufacturer ................ SIEMENS NIXDORF
ID .......................... SNIFC11
Category .................... SYS
Board slot type ............. Embedded
Readable ID ................. Yes
Amperage .................... 10000 milliamps
Overlay name ................ SNIFC11.OVL
Overlay version ............. 4.06



CFG File Extension Version .................... 04.06

System Board Settings
  Diskette Controller ......................... Enabled
  IDE Controller .............................. Disabled

System Board Peripherals
  Serial Port 1 ............................... Disabled
  Serial Port 2 ............................... Disabled
  Parallel Interface .......................... Disabled
  Mouse Interface ............................. Enabled

Memory Equipment
```

```
  BIOS ROM Size ................................ 128Kb
  Base Memory .................................. 640Kb
  Extended Memory .............................. 3327Mb

PCI Configuration
  Mapping INT_A# (Host Bridge 0) ............... AUTO
  Mapping INT_B# (Host Bridge 0) ............... AUTO
  Mapping INT_C# (Host Bridge 0) ............... AUTO
  Mapping INT_D# (Host Bridge 0) ............... AUTO
  Mapping INT_A# (Host Bridge 1) ............... AUTO
  Mapping INT_B# (Host Bridge 1) ............... AUTO
  Mapping INT_C# (Host Bridge 1) ............... AUTO
  Mapping INT_D# (Host Bridge 1) ............... AUTO


  Slot 4 (Host Bridge 0) ....................... Bridge IRQ

  Slot 5 (Host Bridge 0) ....................... Bridge IRQ

  Slot 6 (Host Bridge 1) ....................... Bridge IRQ

  Slot 7 (Host Bridge 1) ....................... Bridge IRQ

  Slot 8 (Host Bridge 1) ....................... Bridge IRQ

  Slot 9 (Host Bridge 1) ....................... Empty

  Onboard VGA-Controller ....................... Enabled



                           Board Information


3Com Fast EtherLink EISA (3C597-TX) Network Adapter

          Slot 2

Manufacturer ................. 3Com Corporation
ID ........................... TCM5970
Category ..................... NET
Board slot type .............. EISA
Readable ID .................. Yes
Skirt ........................ No
Length ....................... 180 millimeters
Amperage ..................... 750 milliamps

Interrupt Request Level ........................ 3
Boot PROM Size ................................. Disabled



                             Used Resources

  Resource                 Slot        Function
  IRQ  0.................. System      Fixed Resources
  IRQ  1.................. System      Fixed Resources
  IRQ  3.................. Slot 2      Interrupt Request Level
  IRQ  6.................. System      Diskette Controller
  IRQ  8.................. System      Fixed Resources
  IRQ 12.................. System      Mouse Interface
  IRQ 13.................. System      Fixed Resources

  DMA  2.................. System      Diskette Controller

  Port 0h - 0FFh.......... System      Fixed Resources
  Port 3C0h - 3CFh........ System      Onboard VGA-Controller
  Port 3D0h - 3DFh........ System      Onboard VGA-Controller
  Port 3F0h - 3F5h........ System      Diskette Controller
  Port 3F7h............... System      Diskette Controller
```

```
   Port 800h - 8FFh........ System      Fixed Resources
   Port 2000h - 200Fh...... Slot 2      I/O Address Range
   Port 46E8h............. System      Onboard VGA-Controller

   Memory
   Address       Amount
 # 0...............640K.... System      Base Memory
   0A0000h..........64K.... System      Onboard VGA-Controller
   0B8000h..........32K.... System      Onboard VGA-Controller
   0C0000h..........32K.... System      Onboard VGA-Controller
 # 0E0000h.........128K.... System      BIOS ROM Size
 # 1M...............63M.... System      Extended Memory 1
 # 64M..............64M.... System      Extended Memory 1
 # 128M.............64M.... System      Extended Memory 1
 # 192M.............64M.... System      Extended Memory 1
 # 256M.............64M.... System      Extended Memory 1
 # 320M.............64M.... System      Extended Memory 1
 # 384M.............64M.... System      Extended Memory 1
 # 448M.............64M.... System      Extended Memory 1
 # 512M.............64M.... System      Extended Memory 2
 # 576M.............64M.... System      Extended Memory 2
 # 640M.............64M.... System      Extended Memory 2
 # 704M.............64M.... System      Extended Memory 2
 # 768M.............64M.... System      Extended Memory 2
 # 832M.............64M.... System      Extended Memory 2
 # 896M.............64M.... System      Extended Memory 2
 # 960M.............64M.... System      Extended Memory 2
 # 1024M............64M.... System      Extended Memory 3
 # 1088M............64M.... System      Extended Memory 3
 # 1152M............64M.... System      Extended Memory 3
 # 1216M............64M.... System      Extended Memory 3
 # 1280M............64M.... System      Extended Memory 3
 # 1344M............64M.... System      Extended Memory 3
 # 1408M............64M.... System      Extended Memory 3


 # 1472M............64M.... System      Extended Memory 3
 # 1536M............64M.... System      Extended Memory 4
 # 1600M............64M.... System      Extended Memory 4
 # 1664M............64M.... System      Extended Memory 4
 # 1728M............64M.... System      Extended Memory 4
 # 1792M............64M.... System      Extended Memory 4
 # 1856M............64M.... System      Extended Memory 4
 # 1920M............64M.... System      Extended Memory 4
 # 1984M............64M.... System      Extended Memory 4

# = Caching


                               Available Resources

 ---IRQs--|---DMAs--|-----ISA I/O Ports---|-Memory Amount---Address----
      4   |    0    |     100h -   3BFh   |        32K      0B0000h
      5   |    1    |     3E0h -   3EFh   |        32K      0C8000h
      7   |    3    |     3F6h            |        64K      0D0000h
    2(9)  |    5    |     3F8h -   400h   |
     10   |    6    |                     |
     11   |    7    |                     |
     14   |         |                     |
     15   |         |                     |


                              System Specifications

 Slot        Slot         Board     Accept   Max     Bus-    Slot
 Name        Type         ID        Skirted  Length  master  Tag(s)
 ----------  ----------   -------   -------  ------  ------  ----------
 Slot 1      EISA         (Empty)   No       341mm   Yes
 Slot 2      EISA         TCM5970   No       341mm   Yes
 Slot 3      EISA         (Empty)   No       341mm   Yes
 Slot 4      EISA         (Empty)   No       341mm   Yes
```

```
Nonvolatile memory ................ 4K




     **********************************************************************
     *        MYLEX Disk Array Controller - Configuration Utility         *
     *                            Version 4.71                            *
     **********************************************************************

     CONFIGURATION INFORMATION OF :
     ==============================

     3 Channel - 15 Target  DAC960PJ   #1    Firmware version 4.00


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 5

     Pack  0 : [0:0]
     Pack  1 : [0:1]
     Pack  2 : [0:2]  [0:3]  [0:4]  [0:5]  [0:6]  [0:8]  [0:9]
     Pack  3 : [1:2]  [1:3]  [1:4]  [1:5]  [1:6]  [1:8]  [1:9]
     Pack  4 : [2:2]  [2:3]  [2:4]  [2:5]  [2:6]  [2:8]  [2:9]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 3

     Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
     =========   =========   ==========   =========   ============
         0         4303  MB        7         4303  MB   Write Thru
         1         4303  MB        7         4303  MB   Write Thru
         2        90363  MB        0        90363  MB   Write Thru



     **********************************************************************
     *        MYLEX Disk Array Controller - Configuration Utility         *
     *                            Version 4.71                            *
     **********************************************************************

     CONFIGURATION INFORMATION OF :
     ==============================

     3 Channel - 15 Target  DAC960PJ   #2    Firmware version 4.00


     PHYSICAL PACK INFORMATION :
     ===========================

     Number of Packs = 3

     Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:4]  [0:5]  [0:6]
     Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:4]  [1:5]  [1:6]
     Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:4]  [2:5]  [2:6]

     SYSTEM DRIVE INFORMATION :
     ==========================

     Number of System Drives = 1

     Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
     =========   =========   ==========   =========   ============
         0        90363  MB        0        90363  MB   Write Thru
```

```
**********************************************************************
*        MYLEX Disk Array Controller - Configuration Utility         *
*                            Version 4.71                            *
**********************************************************************

CONFIGURATION INFORMATION OF :
==============================

3 Channel - 15 Target  DAC960PJ   #3    Firmware version 4.00


PHYSICAL PACK INFORMATION :
===========================

Number of Packs = 3

Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:4]  [0:5]  [0:6]
Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:4]  [1:5]  [1:6]
Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:4]  [2:5]  [2:6]

SYSTEM DRIVE INFORMATION :
==========================

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        90363  MB       0        90363  MB   Write Thru



**********************************************************************
*        MYLEX Disk Array Controller - Configuration Utility         *
*                            Version 4.71                            *
**********************************************************************

CONFIGURATION INFORMATION OF :
==============================

3 Channel - 15 Target  DAC960PJ   #4    Firmware version 4.00


PHYSICAL PACK INFORMATION :
===========================

Number of Packs = 3

Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:4]  [0:5]  [0:6]
Pack  1 : [1:0]  [1:1]  [1:2]  [1:3]  [1:4]  [1:5]  [1:6]
Pack  2 : [2:0]  [2:1]  [2:2]  [2:3]  [2:4]  [2:5]  [2:6]

SYSTEM DRIVE INFORMATION :
==========================

Number of System Drives = 1

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        90363  MB       0        90363  MB   Write Thru



**********************************************************************
*        MYLEX Disk Array Controller - Configuration Utility         *
*                            Version 4.71                            *
**********************************************************************

CONFIGURATION INFORMATION OF :
==============================

3 Channel - 15 Target  DAC960PJ   #5    Firmware version 4.00
```

```
PHYSICAL PACK INFORMATION :
===========================

Number of Packs = 3

Pack  0 : [0:0]  [0:1]  [0:2]  [0:3]  [0:4]  [0:5]  [0:6]  [0:8]
Pack  1 : [2:0]  [2:1]  [2:2]  [2:3]  [2:4]  [2:5]  [2:6]  [2:8]
Pack  2 : [1:0]  [1:1]  [1:2]  [1:3]  [1:4]  [1:5]  [1:6]  [1:8]

SYSTEM DRIVE INFORMATION :
==========================

Number of System Drives = 2

Sys Drv #   Phy. Size   Raid Level   Eff. Size   Write Policy
=========   =========   ==========   =========   ============
    0        127000  MB       0        127000  MB   Write Thru
    1        69464  MB       6        34732  MB   Write Thru
```

Appendix D - Pricing Details

This appendix contains the calculations used to determine the number of disk drives and the number of LAN segments necessary in the priced configuration and the spreadsheet used to determine the price/performance figure.

180 Day Space Calculation

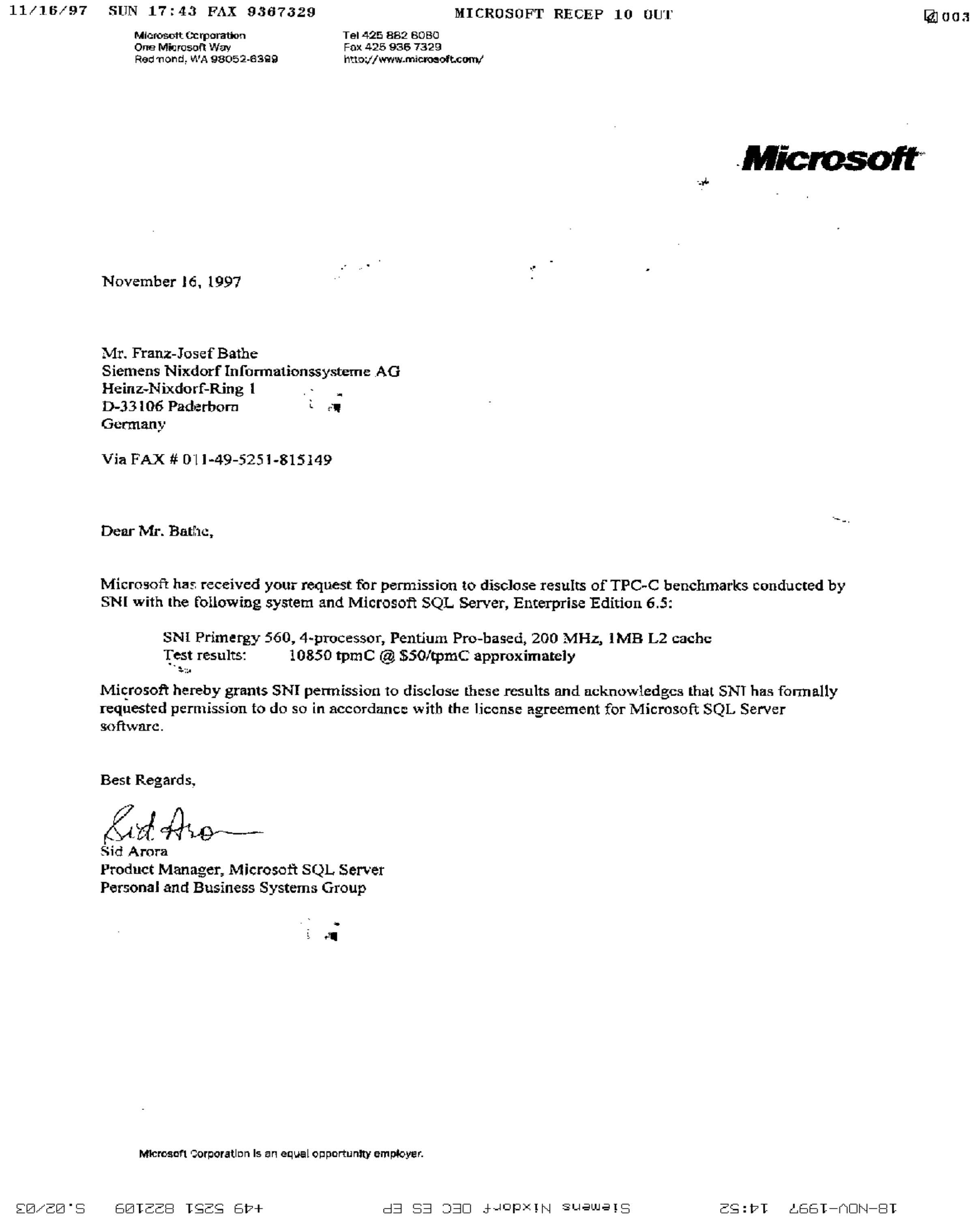
The following worksheet was used to calculate the 180 day space of the system. Note: Numbers are in 2K pages unless otherwise specified

Disk Storage							
Warehouses during measurement:			900				
Warehouses build:			900				
Throughput (tpmC):			10 854.24 tpmC				
Table	Rows	Data 1k pages	Index 1k pages	Overhead	Extra 5%	Total with 5%	
warehouse	900	1 800	12		91	1 903	
district	9 000	18 000	76		904	18 980	
item	100 000	9 100	46		457	9 603	
customer	27 000 000	18 003 600	1 397 330		970 047	20 370 977	
new_order	8 100 000	90 000	548		4 527	95 075	
stock	90 000 000	30 006 000	165 786		1 508 589	31 680 375	
history	27 000 000	1 350 004	0		67 500	1 417 504	
orders	27 000 000	702 000	4 234		35 312	741 546	
order_line	269 999 044	15 008 724	98 104		755 341	15 862 169	
Totals (in MB)		63 661 MB	1 627 MB	0 MB	3 264 MB	68 553 MB	
cs + index	50 831 MB						
ol + index	15 490 MB						
misc + index	2 231 MB						
As loaded	65 288 MB						
As needed for 5%	68 553 MB						
As needed for 8 hours	71 768 MB						
DBspaces	# of segments	size in MB	Total Allocated	Sum	tables here	Space alloced	Space loaded +5%
Master	1	30 MB	30 MB				
Model	1	6 MB	6 MB				
Msdb	1	2 MB	2 MB				
tpcsc	5	10 600 MB	53 000 MB				
tpcol	4	6 000 MB	24 000 MB				
tpcmisc	5	1 000 MB	5 000 MB				
total			82 038 MB			0 MB	0 MB
	in MB						
Dynamic space	16 661 MB		Sum of Data for Order, Order_line and History				
Static space	51 930 MB		Sum of all data, index, bitmap (incl. the rootdbs) + %5 - above dynamic space				
Free space	13 447 MB		Total space allocated to DBMS - dynamic and static				
Daily growth	3 215 MB		(Dynamic space/(W*62.5))*tpmC				
Daily spread	8 625 MB		Free space -1.5 * Daily growth (zero if negative)				
			This can be reconfigured to eliminate daily spread, zero assumed				
180 day space (MB)	630 621 MB		static space + 180 * (daily growth + daily spread)				
180 day space (GB)	615.84 GB						
			NO before	27009000	Log before	653 MB	
			NO after	27759461	Log after	4 733 MB	
			diff	750461		4 080 MB	
			Log usage per NO			5701	Bytes
8 hr log space (GB)	27.66 GB		((increase of log in byte) / (new order transactions)) * tpmC * 60 min * 8 h				
	Space needed	Disk size	Disks Priced	GB			
180 day space	615.84 GB	4.2021 GB	58	243.72 GB			
		8.4795 GB	45	381.58 GB			
Logical logs (w/mirrors)	55.33 GB	8.4795 GB	8	67.84 GB			
OS, file sys, swap	4.0596 GB	4.2021 GB	1	4.20 GB			
Total	675.23 GB		112	697.34 GB			

Price/Performance Spreadsheet *The following detailed worksheet was used to calculate the price/performance of the system.*

Description	Part Number	Third Party		Unit Price	Qty.	Extended Price	5 yr. Maint. Price
		Brand	Pricing				
Server Hardware							
Base System	S26361-K412-V392			3 513 $	1	3 513 $	
PSU-Modul	S26113-E379-E10			179 $	2	359 $	
1. CPU Module	S26361-F1308-E1			455 $	1	455 $	
2. CPU Modul	S26361-F1308-E10			455 $	1	455 $	
Pentium Pro 200MHz/1MB	S26361-F1329-E202			5 747 $	4	22 989 $	
Kit for PPro 1MB SLC	S26361-F1718-E1			28 $	1	28 $	
Memory 1 GB (4x256) DIMM	S26361-F1307-E26			17 832 $	3	53 497 $	
Memory 256 MB (4x64) DIMM	S26361-F1307-E23			3 090 $	1	3 090 $	
Mylex Disc Array Controller PCI incl. 10% spare	S26361-F1779-E1			1 375 $	7	9 623 $	
Connectors for Disk Cabinets	S26361-F1222-E21			69 $	5	345 $	
Fast-Ether-Express-Pro 100Mbit (PCI)	S26361-F1465-E501			101 $	1	101 $	
Keyboard	S26381-K252-L165			39 $	1	39 $	
Country Pack	S26361-F1290-B173			37 $	1	37 $	
Sum Primergy 560							5 805 $
Monitor MCM 1405 ND	S26361-K449-V150			253 $	1	253 $	76 $
POD-SE/W disk cabinet incl. 10% spare	S26361-K377-V291			1 655 $	17	28 138 $	
HD W-SCSI 4GB hot plug for POD-Sew incl. 10% s	S26361-F1145-E140			943 $	65	61 264 $	
HD W-SCSI 9GB hot plug for POD-Sew incl. 10% s	S26361-F1145-E181			1 664 $	59	98 198 $	
CD-ROM 8x for POD-Sew	S26361-F726-E75			207 $	1	207 $	62 $
W-SCSI Cable UHD-HD incl. 10% spare	T26139-Y2549-V1			88 $	11	971 $	
W-SCSI Cable HD-HD incl. 10% spare	T26139-Y2527-M1			69 $	6	414 $	
2. Bridge Connector incl. 10% spare	S26361-F1148-L21			44 $	17	743 $	
					Subtotal	284 716 $	5 943 $
Server Software							
Microsoft NT-Server 4.0, Enterprise Edition	Microsoft	MS	1	3 999 $	1	3 999 $	
MS SQL-Server 6.5 Enterp.Edition unlim. license	Microsoft	MS	1	28 999 $	1	28 999 $	
					Subtotal	32 998 $	10 475 $
Client Hardware							
Primergy 160	S26361-K423-V744			3 191 $	6	19 145 $	
Keyboard	S26381-K252-V165			39 $	6	234 $	
Country Pack	S26361-F1454-B233			37 $	6	221 $	
Memory 64 MB EDO DIMM	S26361-F1514-E504			690 $	18	12 414 $	
Fast-Ether-Express-Pro 100Mbit (PCI)	S26361-F1465-E501			101 $	18	1 821 $	
Sum Primergy 160							23 520 $
Monitor MCM 1405 ND	S26361-K449-V150			253 $	6	1 517 $	455 $
					Subtotal	35 352 $	23 975 $
Client Software							
NT-Server 4.0		MS	1	809 $	6	4 854 $	
MS SQL-Server Prog. Toolkit		MS	1	499 $	1	499 $	
Open UTM	U11421-C10			973 $	6	5 838 $	8 820 $
MS Visual C++		MS	1	499 $	1	499 $	
					Subtotal	11 690 $	8 820 $
User Connectivity							
ATI 24 PORT HUB incl. 10% spare	AT-3024TR			160 $	413	66 080 $	
Fast Ethernet Hub 8*100 incl. 10% spare	AT-908TX-20			320 $	3	960 $	
					Subtotal	67 040 $	
					Total	431 796 $	49 213 $

Appendix E - Price Quotations



11/16/97 SUN 17:43 FAX 9367329 MICROSOFT RECEP 10 OUT ☑003

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

November 16, 1997

Mr. Franz-Josef Bathe
Siemens Nixdorf Informationssysteme AG
Heinz-Nixdorf-Ring 1
D-33106 Paderborn
Germany

Via FAX # 011-49-5251-815149

Dear Mr. Bathe,

Microsoft has received your request for permission to disclose results of TPC-C benchmarks conducted by SNI with the following system and Microsoft SQL Server, Enterprise Edition 6.5:

SNI Primergy 560, 4-processor, Pentium Pro-based, 200 MHz, 1MB L2 cache
Test results: 10850 tpmC @ $50/tpmC approximately

Microsoft hereby grants SNI permission to disclose these results and acknowledges that SNI has formally requested permission to do so in accordance with the license agreement for Microsoft SQL Server software.

Best Regards,

Sid Aro

Sid Arora
Product Manager, Microsoft SQL Server
Personal and Business Systems Group

Microsoft Corporation is an equal opportunity employer.

18-NOV-1997 14:52 Siemens Nixdorf OEC ES EP +49 5251 822109 S.02/03

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

November 16, 1997

Mr. Franz-Josef Bathe
Siemens Nixdorf Informationssysteme AG
Heinz-Nixdorf-Ring 1
D-33106 Paderborn
Germany

Via FAX # 011-49-5251-815149

Dear Mr. Bathe,

Here is the information you requested regarding US pricing of certain Microsoft products:

Microsoft SQL Server, Enterprise Edition 6.5, unlimited user license	$28999
Microsoft Windows NT Server, Enterprise Edition 4.0, incl 25 CALs	$3999
Windows NT Server 4.0, incl 5 CALs	$809
Microsoft SQL Workstation (includes programmers toolkit)	$499
Visual C++ 32-bit edition (subscription)	$499
5-yr maintenance for above software @ $2095/yr	$10475

The prices quoted above are valid for the next 60 days. Please let me know if I can be of any further assistance.

Sincerely,

Sid Arora

Sid Arora
Product Manager, Microsoft SQL Server
Personal and Business Systems Group

Microsoft Corporation is an equal opportunity employer.

Allied Telesyn

Allied Telesyn International GmbH · Postfach 270 222 · D 13472 Berlin

Siemens Nixdorf Informationssysteme AG
OEC HES PM 4
Herrn Seidel
Heinz-Nixdorf-Ring 1

33094 Paderborn

Berlin, 10. November 1997

Repeater AT-3024SL

Allied Telesyn International, Inc. is pleased to confirm that the following product is available to all SNI Customers for the listed US price list.

Product	**Description**	**Special Purchase Price**	
AT-3024SL	Multiport Repeater 1*AUI/1*BNC 24 shielded TP-Ports, unmanaged, Slimline	for 410 pcs	$ 160
AT-908TX-20	Fast Ethernet Hub 12*100BaseTX/RJ45 Stack Option		$ 320

Best Regards

Allied Telesyn International GmbH

Antje Popp
Antje Popp
Account Executive

Allied Telesyn International GmbH
Wittestraße 30N · D 13509 Berlin · Tel. (+49-30) 435 90 0-0 · Fax (+49-30) 435 70 650 (=Allied), (+49-30) 432 6163
Geschäftsstelle Süd: Gute Änger 13 · D 85356 Freising · Tel. (+49-8161) 99 06-0 · Fax (+49-8161) 99 06-22
Geschäftsführer: Dr. Rüdiger Meisenburg · Sitz: Berlin · HRB 36522 · Amtsgericht Charlottenburg · UStID-Nr.: DE 136609004
Bankverbindung: Berliner Bank AG · BLZ 100 200 00 · DM-Kto.-Nr. 17 68 17 66 00 · US$-Kto.-Nr. 17 68 17 66 01

Appendix F - Attestation Letter

Information Paradigm

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL
Certified Auditor

Sponsor: Ingo Schulte
Manager, Benchmark Center
Siemens Nixdorf Informationssysteme AG
Heinz-Nixdorf-Ring 1
33106 Paderborn
Germany

February 12, 1997

I remotely verified the TPC Benchmark™ C performance of the following Client Server configuration:

Platform:	Primergy 560 c/s
Operating system:	Windows NT 4.0
Database Manager:	Microsoft SQL Server 6.5
Other Software:	Microsoft Internet Connector

The results were:

CPU's Speed	Memory	Disks	NewOrder 90% Response Time	tpmC
Server: Primergy 560				
4 x Pentium Pro (200 MHz)	2048 MB	71 x 4 GB 21 x 9 GB	2.31 Seconds	7063.07
(5) Clients: Primergy 160 (Specification for each)				
1 x Pentium (200 MHz)	128 MB	1 x 2 GB	n/a	n/a

In my opinion, these performance results were produced in compliance with the TPC requirements for Revision 3.2.3 of the benchmark. The following verification items were given special attention:

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

- The ACID properties were met
- Input data was generated according to the specified percentages
- The transaction cycle times included the required keying and think times
- The reported response times were correctly measured.
- At least 90% of all delivery transactions met the 80 Second completion time limit
- All 90% response times were under the specified maximums
- The measurement interval was representative of steady state conditions
- The reported measurement interval was 30 minutes (1800 seconds).
- One checkpoint was taken during the measurement interval
- Measurement repeatability was verified
- The 180 day storage requirement was correctly computed
- The system pricing was verified for major components and maintenance

Additional Audit Notes:

None.

Respectfully Yours,

François Raab
President

Primergy 560