NEC

TPC Benchmark™ H
Full Disclosure Report

NEC Express5800/1320Xe (32 SMP)

with Microsoft SQL Server 2005
Enterprise Itanium Edition
and
Microsoft® Windows® Server 2003, Datacenter Edition
for 64-bit Itanium-based Systems Service Pack1

First Edition
Submitted for Review
November 7, 2005

NEC Corporation (NEC) , the Sponsor of this benchmark test, believes that the technical, pricing and discounting information in this document is accurate as of its publication date. The performance information in this document is for guidance only. System performance is highly dependent on many factors including system hardware, system and user software, and user-application characteristics. Customer applications must be carefully evaluated before estimating performance. NEC does not warrant or represent that a user can or will achieve similar performance as expressed in this document.

THE TERMS AND CONDITIONS GOVERNING THE SALE OF NEC HARDWARE PRODUCTS AND THE LICENSING OF NEC SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN NEC AND ITS CUSTOMERS. NO REPRESENTATION OR OTHER AFFIRMATION OF FACT CONTAINED IN THIS DOCUMENT INCLUDING BUT NOT LIMITED TO STATEMENTS REGARDING PRICE, CAPACITY, RESPONSE-TIME PERFORMANCE, SUITABILITY FOR USE, OR PERFORMANCE OF PRODUCTS DESCRIBED HEREIN SHALL BE DEEMED TO BE A WARRANTY BY NEC FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF NEC WHATSOEVER.

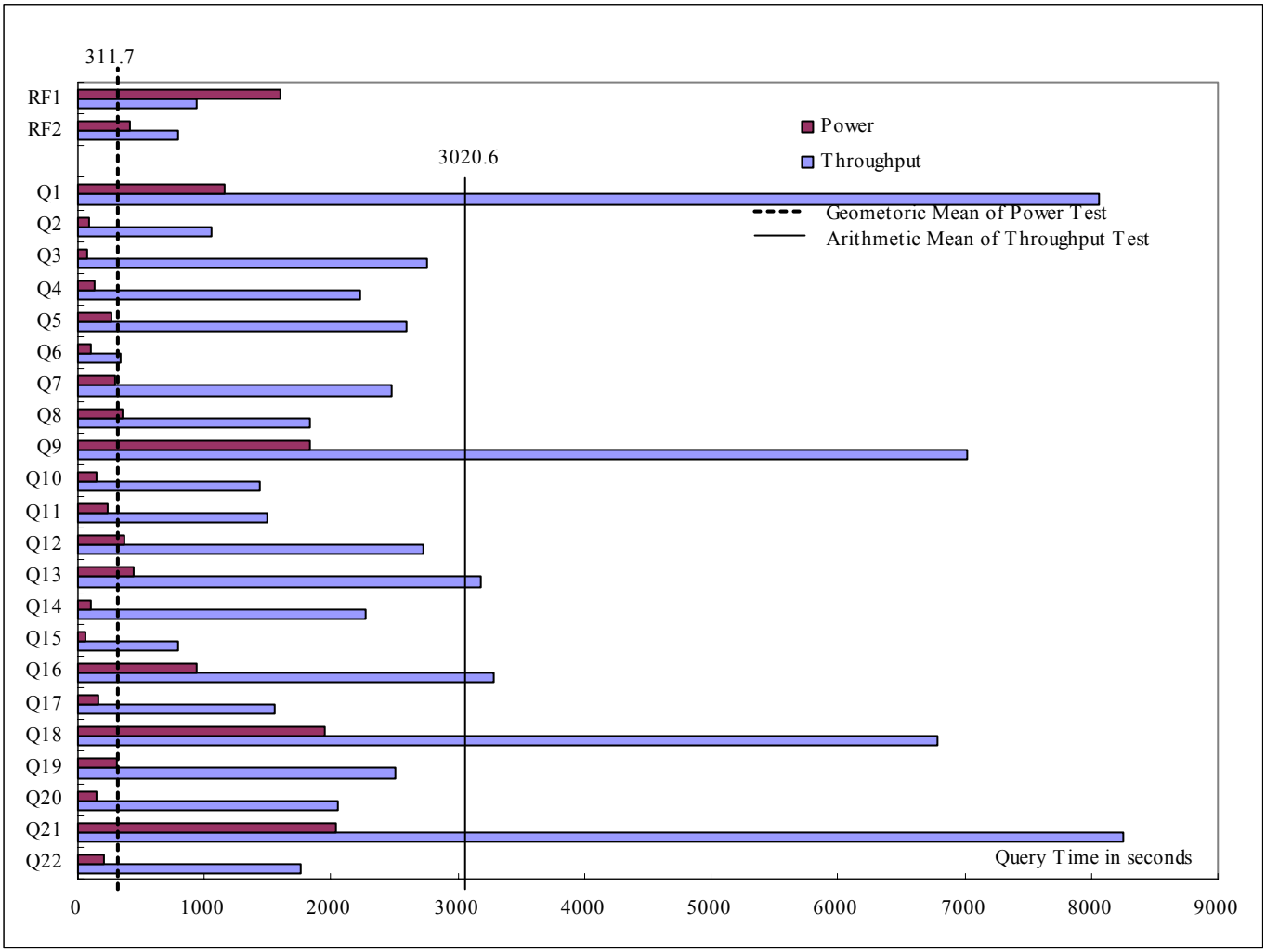
NEC assumes no responsibility for any errors that may appear in this document. NEC reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult NEC to determine whether any such changes have been made.

Printed in USA, 2005

NEC	NEC Express5800/1320Xe (32 SMP)	TPC-H Rev. 2.3.0 Report Date November 7, 2005

Total System Cost	Composite Query per Hour Metric	Price/Performance
$1,888,276 (USD)	28,390.1 QphH@3000GB	$66.52 / QphH@3000GB

Database size	Database Manager	Operating System	Other Software	Availability Date
3000 GB	Microsoft SQL Server 2005 Enterprise Itanium Edition	Microsoft® Windows® Server 2003, Datacenter Edition for 64-bit Itanium-based Systems SP1	Microsoft® Visual Studio .NET 2003	May 5, 2006



Database Load Time = 36:04:45	Load include backup:Y	Disk Size/Database Size = 11.66
RAID(Base table only):N	RAID(Base tables and auxiliary data structures):N	RAID(All):N

System Configuration:

Total Processors/Cores/Threads	32/32/32	Intel® Itanium® 2 1.6GHz 9MB L3 cache
Memory	256GB	Main Memory
Disk Controllers	1	QLogic™ QLA2340
	22	QLogic™ QLA2342
	1	SCSI HBA
Disk Drives	1050	36GB FC
	1	36GB SCSI
Total Disk Storage		34,983 GB

NEC Express5800/1320Xe (32 SMP)

TPC-H Rev. 2.3.0

Report Date
November 7, 2005

Description	Part Number	Third Party Brand	Pricing	Unit Price	Qty	Extended Price	3-yr Mnt. Price
Server Hardware							
Express5800/1320Xe system (32way model)	850200718	NEC	1	1,133,250	1	1,133,250	223,520
Base Original 32(Express5800/1320Xe base system)					1	-	-
Itanium2 1.6GHz/9M cache					32	-	-
Cell Card Original					8	-	-
4GB Memory Option (4 x 1024MB DIMMs)					64	-	-
Core PCI box					1	-	-
Base PCI box					3	-	-
PCI Cable					8	-	-
Expansion Cabinet					1	-	-
36GB Ultra 320/m SCSI HDD 10000RPM					1	-	-
10/100/1000Mb Either card					1	-	-
Ultra160 SCSI card					1	-	-
Windows Server 2003, Data Center Edition					32	-	-
NEC Express5800/120Rg-2(for System Management)							
120Rg-2 base rack svr, one XEON processor, 3.0 GHz/1MB cache	850174001	NEC	1	3,799	1	3,799	-
1GB (2 x 512MB) interleaved memory DDR333 SDRAM	062-02466-000	NEC	1	999	1	999	-
36.4GB SCSI HDD (1")	062-02012-000	NEC	1	399	1	399	-
CD-ROM, 2 x On-board LAN, KB/MS	Included	NEC	1		1	-	-
3 years of warranty service to 4-hour response, 7x24	DE-0000-1095-7244	NEC	1	1,599	1	-	1,599
CTX CRT VL5103 VL510 15	VL5103-BX	CTX	3	94	2	188	-
FC HBA QLA2340 (+10% spares)	408081	QLogic	4	948	3	2,843	-
FC HBA QLA2342 (+10% spares)	QLA2342-CK-BK	QLogic	3	1,395	25	34,875	-
					Subtotal	1,176,353	225,119
Disk Subsystem							
NEC Storage S2400 Base Model 2 RAID_Cntrl, 2GB Cache, 1DE, w/o drives	850180001	NEC	1	53,000	1	53,000	-
S2400 Redundant Disk Expansion box with cables (+10% spares)	062-02616-000	NEC	1	6,200	5	31,000	-
36GB 15K rpm FC HDD for S2400 (+10% spares)	062-02620-000	NEC	1	790	66	52,140	-
NEC Storage S2300 Base Model 2 RAID_Cntrl, 1GB Cache, 1DE, w/o drives	850171001	NEC	1	33,999	22	747,978	-
S2300 Redundant Disk Expansion box with cables (+10% spares)	062-02402-000	NEC	1	7,999	49	391,951	-
36GB 15K rpm FC HDD for S2300 (+10% spares)	062-02431-000	NEC	1	579	1089	630,531	-
3 yr Telephone Upgrade Hdw & Storage	IS-THSU-1095-0724	NEC	1	1,499	23	-	34,477
42U Rackframe	050-01790-000	NEC	1	1,799	7	12,593	-
FC Cable 10M LC-LC (+10% spares)	062-02304-000	NEC	1	200	50	10,000	-
					Subtotal	1,929,193	34,477
Server Software							
Microsoft SQL Server 2005 Enterprise Itanium Edition	810-04793	Micsoroft	2	8,487	1	8,487	-
Microsoft SQL Server 2005 Client Licenses	359-01911	Microsoft	2	156	65	10,140	-
Microsoft Windows 2003 Standard Edition	P73-00295	Microsoft	2	719	1	719	-
Visual Studio .NET 2003 Professional	659-00390	Microsoft	2	799	1	799	
Microsot Problem Resolution Services		Microsoft	2	245	1	-	245
					Subtotal	20,145	245
					TOTAL	3,125,691	259,841
NEC brand total(Pricing 1-NEC)						3,067,640	259,596
NEC brand Large Purchase Cash Prepay Discount	-45%					-1,380,438	-116,818

Notes:
Pricing: 1-NEC 2-Microsoft 3-Compuview Microsystems 4-CDW

3-Yr. Cost of Ownership: **$1,888,276**
QphH@3TB: **28390.1**
$ / QphH@3TB: **$66.52**

Results and methodology audited by Francois Raab of InfoSizing, Inc. (www.sizing.com)

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

Numerical Quantities Summary

Measurement Results

Scale Factor	3000 GB
Total Data Storage / Database Size	11.66
Start of Database Load	10/24/2005 12:41:40
End of Database Load	10/26/2005 00:46:25
Database Load Time	36:04:45
Query Streams for Throughput Test	8
TPC-H Power	34652.8
TPC-H Throughput	23259.3
Composite Query per Hour Rating (QphH@3000GB)	28390.1
Total System Price Over 3 Years	$1,888,276
TPC-H Price Performance Metric	$66.52

Measurement Intervals

Measurement Interval in Throughput Test (Ts)	81722 seconds

Duration of Stream Execution:

	Seed	Query Start Date/Time Query End Date/Time	RF1 Start Date/Time RF1 End Date/Time	RF2 Start Date/Time RF2 End Date/Time	Query Duration RF Duration
Stream 0	1026004625	10/27/2005 13:29:41 10/27/2005 16:36:59	10/27/2005 13:03:10 10/27/2005 13:29:40	10/27/2005 16:40:00 10/27/2005 16:46:55	
Stream 1	1026004626	10/27/2005 16:46:58 10/28/2005 11:15:30	10/28/2005 11:34:58 10/28/2005 11:46:16	10/28/2005 11:46:21 10/28/2005 11:56:01	18:28:32 0:20:59
Stream 2	1026004627	10/27/2005 16:46:58 10/28/2005 11:29:36	10/28/2005 11:56:07 10/28/2005 12:09:05	10/28/2005 12:09:08 10/28/2005 12:21:11	18:42:38 0:25:01
Stream 3	1026004628	10/27/2005 16:46:59 10/28/2005 11:22:47	10/28/2005 12:21:20 10/28/2005 12:35:26	10/28/2005 12:35:31 10/28/2005 12:47:50	18:35:48 0:26:25
Stream 4	1026004629	10/27/2005 16:47:00 10/28/2005 11:27:56	10/28/2005 12:48:01 10/28/2005 13:02:48	10/28/2005 13:02:54 10/28/2005 13:15:44	18:40:55 0:27:36
Stream 5	1026004630	10/27/2005 16:47:01 10/28/2005 10:05:04	10/28/2005 13:15:58 10/28/2005 13:31:54	10/28/2005 13:32:00 10/28/2005 13:45:15	17:18:03 0:29:11
Stream 6	1026004631	10/27/2005 16:47:03 10/28/2005 11:10:12	10/28/2005 13:45:39 10/28/2005 14:02:42	10/28/2005 14:02:49 10/28/2005 14:17:14	18:23:09 0:31:27
Stream 7	1026004632	10/27/2005 16:47:05 10/28/2005 11:30:38	10/28/2005 14:17:57 10/28/2005 14:36:23	10/28/2005 14:36:48 10/28/2005 14:51:53	18:43:33 0:33:31
Stream 8	1026004633	10/27/2005 16:47:07 10/28/2005 11:34:58	10/28/2005 14:52:41 10/28/2005 15:12:58	10/28/2005 15:13:32 10/28/2005 15:29:00	18:47:52 0:35:44

NEC

NEC Express5800/1320Xe (32 SMP)

TPC-H Rev. 2.3.0

Reported Date
November 7, 2005

TPC-H Timing Intervals (in seconds):

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream 0	1164.2	83.8	80.0	129.6	259.7	106.2	295.3	354.8
Stream 1	6453.1	469.9	846.3	5324.2	2896.3	278.0	2051.2	1213.9
Stream 2	7918.3	1338.6	1199.0	1301.7	2890.0	560.3	2804.6	871.1
Stream 3	7127.4	211.3	4332.9	719.7	5635.7	281.5	2984.4	991.8
Stream 4	7821.6	306.7	3626.6	1360.9	867.7	193.0	1054.7	2468.1
Stream 5	9609.0	2361.3	2292.8	887.4	1781.6	347.4	4188.8	3269.1
Stream 6	7766.5	1013.4	5247.6	3938.5	2570.1	392.9	1950.9	1977.2
Stream 7	7419.5	1935.6	2965.3	3146.7	1020.2	330.3	2025.1	1379.6
Stream 8	10395.0	843.9	1535.3	1170.6	3132.0	284.9	2742.5	2508.8
Min Qi	6453.1	211.3	846.3	719.7	867.7	193.0	1054.7	871.1
Max Qi	10395.0	2361.3	5247.6	5324.2	5635.7	560.3	4188.8	3269.1
Avg Qi	8063.8	1060.1	2755.7	2231.2	2599.2	333.5	2475.3	1835.0
Query	**Q9**	**Q10**	**Q11**	**Q12**	**Q13**	**Q14**	**Q15**	**Q16**
Stream 0	1832.6	141.3	236.3	373.7	445.5	109.8	58.2	933.6
Stream 1	11824.8	1552.7	2697.0	1922.3	3708.8	867.5	1646.1	2614.4
Stream 2	9556.6	1701.9	1012.6	2767.6	2606.7	4574.4	577.8	2578.9
Stream 3	5279.1	1567.9	1917.7	3769.7	3037.5	2013.7	345.3	3846.7
Stream 4	4881.8	1744.6	1800.2	3961.8	2183.7	1232.8	1222.3	3171.3
Stream 5	4209.3	952.9	1914.2	3160.5	3104.8	686.2	754.3	3478.4
Stream 6	5077.0	1340.2	1318.3	2551.1	4019.8	1224.7	511.5	3289.6
Stream 7	7971.5	1290.3	1041.6	614.7	2801.1	5224.7	169.4	3311.3
Stream 8	7391.7	1304.7	215.8	3042.3	4015.3	2347.5	1078.8	4012.4
Min Qi	4209.3	952.9	215.8	614.7	2183.7	686.2	169.4	2578.9
Max Qi	11824.8	1744.6	2697.0	3961.8	4019.8	5224.7	1646.1	4012.4
Avg Qi	7024.0	1431.9	1489.7	2723.8	3184.7	2271.4	788.2	3287.9
Query	**Q17**	**Q18**	**Q19**	**Q20**	**Q21**	**Q22**	**RF1**	**RF2**
Stream 0	163.6	1956.0	312.2	148.2	2032.9	200.0	1590.7	414.7
Stream 1	1081.2	6771.5	1357.3	3629.8	6085.8	1219.6	678.1	580.3
Stream 2	1712.6	6538.0	2925.5	1556.4	8008.7	2356.5	778.2	722.8
Stream 3	2330.0	6666.0	1551.5	543.2	9373.6	2421.0	846.7	738.6
Stream 4	1339.7	6328.3	3455.4	4781.0	12076.4	1377.0	886.2	769.7
Stream 5	1867.7	7542.2	1893.6	524.2	6080.6	1376.1	956.2	794.4
Stream 6	1171.4	7371.8	2025.8	2344.0	7627.6	1459.0	1023.2	864.2
Stream 7	538.7	6367.3	4380.3	2298.4	9645.8	1535.8	1106.0	905.0
Stream 8	2353.9	6714.3	2451.7	686.8	7108.0	2335.5	1216.8	927.7
Min Qi	538.7	6328.3	1357.3	524.2	6080.6	1219.6	678.1	580.3
Max Qi	2353.9	7542.2	4380.3	4781.0	12076.4	2421.0	1216.8	927.7
Avg Qi	1549.4	6787.4	2505.1	2045.5	8250.8	1760.1	936.4	787.8

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor: Jun Suzuki
NEC Corporation
1-10 Nisshincho
Fuchu-City, Tokyo 183-8501,
Japan

November 1, 2005

I verified the TPC Benchmark™ H performance of the following configuration:

Platform: **NEC Express5800/1320Xe (32 SMP)**
Database Manager: **Microsoft SQL Server 2005 Enterprise Itanium Edition**
Operating System: **Microsoft Windows Server 2003 Datacenter Edition for 64-bit**

The results were:

CPU (Speed)	Memory	Disks	**QphH@3000GB**
NEC Express5800/1320Xe			
32 x Intel Itanium 2 (1.6 GHz)	9 MB L3 256 GB Main	1050 x 36 GB FC 1 x 36 GB SCSI	**28,390.1**

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

- The database records were defined with the proper layout and size
- The database population was generated using DBGEN
- The database was properly scaled to 3,000GB and populated accordingly
- The compliance of the database auxiliary data structures was verified
- The database load time was correctly measured and reported

- The required ACID properties were verified and met
- The query input variables were generated by QGEN
- The query text was produced using minor modifications
- The execution of the queries against the SF1 database produced compliant answers
- A compliant implementation specific layer was used to drive the tests
- The throughput tests involved 8 query streams
- The ratio between the longest and the shortest query was such that no query timing was adjusted
- The execution times for queries and refresh functions were correctly measured and reported
- The repeatability of the measured results was verified
- The required amount of database log was configured
- The system pricing was verified for major components and maintenance
- The major pages from the FDR were verified for accuracy

Additional Audit Notes:

None.

Respectfully Yours,

François Raab
President

Preface

Document Overview

This report documents the methodology and results of the TPC Benchmark™ H (TPC-H) test conducted on the NEC Express5800/1320Xe using Microsoft SQL Server 2005 Enterprise Itanium Edition, in conformance with the requirements of the TPC Benchmark™H Standard Specification Revision 2.3.0. The tests documented in this report were sponsored by NEC Corporation. The operating system used for the benchmark was Microsoft Windows Server 2003, Datacenter Edition for 64-bit Itanium-based Systems, Service Pack 1. The Transaction Processing Performance Council (TPC) developed the TPC-H Benchmark. The TPC Benchmark™ H Standard represents an effort by NEC Corporation and other members of the Transaction Processing Performance Council (TPC) to create an industry-wide benchmark for evaluating the performance and price/performance of decision support systems, and to disseminate objective, verifiable performance data to the data processing industry. A certified audit of these measurements and the reported results was performed by Francois Raab of InfoSizing, Inc.(Colorado Springs, CO). He has verified compliance with the relevant TPC Benchmark™ H specifications; audited the benchmark configuration, environment, and methodology used to produce and validate the test results; and audited the pricing model used to calculate the price/ performance. The auditor's letter of attestation is attached to the Executive Summary and precedes this section.

TPC Benchmark™ H Overview

The TPC Benchmark™H (TPC-H) is a decision support benchmark. It consists of a suite of business oriented ad-hoc queries and concurrent updates. The queries and the data populating the database have been chosen to have broad industry-wide relevance while maintaining a sufficient degree of ease of implementation. This benchmark illustrates decision support systems that:

- Examine large volumes of data;
- Execute queries with a high degree of complexity;
- Give answers to critical business questions.

TPC-H evaluates the performance of various decision support systems by the execution of sets of queries against a standard database under controlled conditions. The TPC-H queries:

- Give answers to real-world business questions;
- Simulate generated ad-hoc queries (e.g., via a point and click GUI interface);
- Are far more complex than most OLTP transactions;
- Include a rich breadth of operators and selectivity constraints;
- Generate intensive activity on the part of the database server component of the system under test;
- Are executed against a database complying to specific population and scaling requirements;
- Are implemented with constraints derived from staying closely synchronized with an on-line production database.

The TPC-H operations are modeled as follows:

- The database is continuously available 24 hours a day, 7 days a week, for ad-hoc queries from multiple end users and updates against all tables, except possibly during infrequent (e.g., once a month) maintenance sessions;
- The TPC-H database tracks, possibly with some delay, the state of the OLTP database through on-going updates which batch together a number of modifications impacting some part of the decision support database;
- Due to the world-wide nature of the business data stored in the TPC-H database, the queries and the updates may be executed against the database at any time, especially in relation to each other. In addition, this mix of queries and updates is subject to specific ACIDity requirements, since queries and updates may execute concurrently;
- To achieve the optimal compromise between performance and operational requirements the database administrator can set, once and for all, the locking levels and the concurrent scheduling rules for queries and updates.

The minimum database required to run the benchmark holds business data from 10,000 suppliers. It contains almost ten million rows representing a raw storage capacity of about 1 gigabyte. Compliant benchmark implementations may also use one of the larger permissible database populations (e.g., 1000 gigabytes), as defined in Clause 4.1.3.

The performance metric reported by TPC-H is called the TPC-H Composite Query-per-Hour Performance Metric (QphH@Size), and reflects multiple aspects of the capability of the system to process queries. These aspects include the selected database size against which the queries are executed, the query processing power when queries are submitted by a single stream, and the query throughput when queries are submitted by multiple concurrent users. The TPC-H Price/Performance metric is expressed as $/QphH@Size. To be compliant with the TPC-H standard, all references to TPCH results for a given configuration must include all required reporting components. *The TPC believes that comparisons of TPC-H results measured against different database sizes are misleading and discourages such comparisons.*

The TPC-H database must be implemented using a commercially available database management system (DBMS) and the queries executed via an interface using dynamic SQL. The specification provides for variants of SQL, as implementers are not required to have implemented a specific SQL standard in full.

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

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

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

Benchmark sponsors are permitted several possible system designs, provided that they adhere to the model described in Clause 6. A full disclosure report (FDR) of the implementation details, as specified in Clause 8, must be made available along with the reported results.

General Implementation Guidelines

The purpose of TPC benchmarks is to provide relevant, objective performance data to industry users. To achieve that purpose, TPC benchmark specifications require that benchmark tests be implemented with systems, products, technologies and pricing that:

- Are generally available to users;
- Are relevant to the market segment that the individual TPC benchmark models or represents (e.g. TPC-H models and represents complex, high data volume, decision support environments);
- Would plausibly be implemented by a significant number of users in the market segment the benchmark models or represents.

Related Product Information

The TPC Benchmark™ H Standard requires that test sponsors provide a Full Disclosure Report in addition to published results. You can obtain copies of the test results as well as additional copies of this full disclosure report by sending a request to the following address:

Transaction Processing Performance Council
Presidio of San Francisco
Building 572B (surface)
P.O. Box 29920 (mail) San Francisco, CA 94129-0920
Voice: 415-561-6272
Fax: 415-561-6120
Email: info@tpc.org

1. General Items

1.1 Benchmark Sponsor

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

NEC Corporation is the sponsor of this TPC-H benchmark. The benchmark implementation was developed by Microsoft and NEC. The benchmark was conducted at NEC in Fuchu, Tokyo.

1.2 Parameter Settings

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

- *Data Base tuning options;*
- *Optimizer/Query execution options;*
- *Query Processing tool/language configuration parameters;*
- *Recovery/commit options;*
- *Consistency/locking options;*
- *Operating system and configuration parameters;*
- *Configuration parameters and options for any other software component incorporated into the pricing structure;*
- *Compiler optimization options.*

Comment 1*: In the event that some parameters and options are set multiple times, it must be easily discernible by an interested reader when the parameter or option was modified and what new value it received each time.The TPC Executive Summary Statement is included at the beginning of this report.*

Comment 2*: This requirement can be satisfied by providing a full list of all parameters and options, as long as all those that have been modified from their default values have been clearly identified and these parameters and options are only set once*.

Details of system and database configurations and parameters are provided in Appendixes A and B.

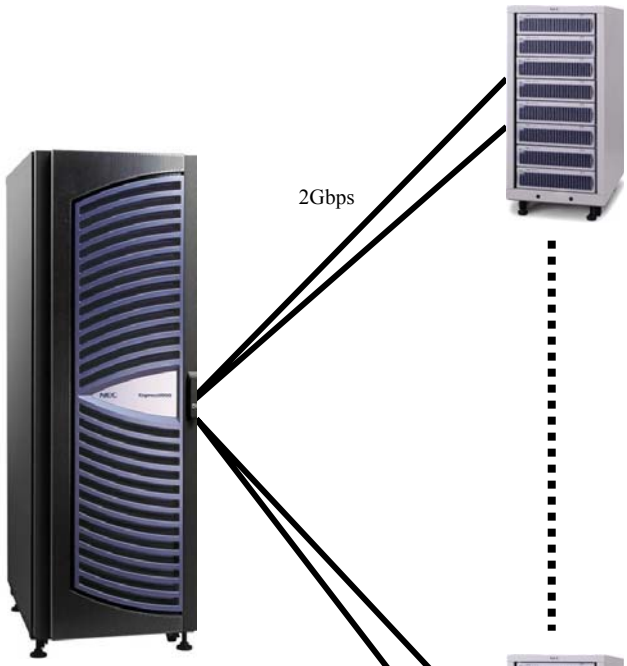
1.3 Configuration Diagrams

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, work stations, terminals, etc., that were physically used in the test or are incorporated into the pricing structure;*
- *Type and run-time execution location of software components (e.g., DBMS, query processing tools/languages, middle-ware components, software drivers, etc.).*

The configuration diagram for the tested and priced system is provided on the following page.

Figure1.1: Express5800/1320Xe, Tested and Priced Configurations



NEC Express5800/1320Xe
32 x Intel® Itanium® 2 1.6GHz
9MB L3 cache
256GB Memory
1 x QLogic™ QLA2340
22 x QLogic™ QLA2342
1 x SCSI HBA
1 x Gbit Ether NIC
1 x 36GB Disk

NEC Storage S2300
990 x 36GB Disks
22 x FC Raid Controller Unit

NEC Storage S2400
60 x 36GB Disks
1 x FC Raid Controller Unit

2 Clause 1 : Logical Database Design

2.1 Table Definitions

Listings must be provided for all table definition statements and all other statements used to setup the test and qualification databases.

Appendix B contains the scripts that create the tables and indexes for the TPC-H database.

2.2 Database Organization

The physical organization of tables and indices, within the test and qualification databases, must be disclosed. If the column ordering of any table is different from that specified in Clause 1.4, it must be noted.

Clustered indexes were used. See Appendix B, which contains the database and table creation statements.

2.3 Horizontal Partitioning

Horizontal partitioning of base tables or auxiliary structures created by database directives is allowed. Groups of rows from a table or auxiliary structure may be assigned to different files, disks, or areas. If this assignment is a function of data in the table or auxiliary structure, the assignment must be based on the value of a partitioning field. A partitioning field must be one and only one of the following:

- *A primary*
- *A foreign*
- *A single date column*

Some partitioning schemes require the use of directives that specify explicit values for the partitioning field. If such directives are used they must satisfy the following conditions:

- *They may not rely on any knowledge of the data stored in the table except the minimum and maximum values of columns used for the partitioning field.*
- *Within the limitations of integer division, they must define each partition to accept an equal portion of the range between the minimum and maximum values of the partitioning column(s).*
- *The directives must allow the insertion of values of the partitioning column(s) outside the range covered by the minimum and maximum values.*

Multiple-level partitioning of base tables or auxiliary structures is allowed only if each level of partitioning satisfies the conditions stated above and each level references only one partitioning field as defined above. If implemented, the details of such partitioning must be disclosed.

Horizontal partitioning was not used. See Appendix B, which contains the database and table creation statements.

2.4 Vertical Partitioning

Vertical partitioning of tables is not allowed. For example, groups of columns of one row shall not be assigned to files, disks, or areas different from those storing the other columns of that row. The row must be processed as an atomic series of contiguous columns.

***Comment**: The effect of vertical partitioning is to reduce the effective row size accessed by the system. Given the synthetic nature of this benchmark, the effect of vertical partitioning is achieved by the choice of row sizes. No further vertical partitioning of the data set is allowed. Specifically, the above Clause prohibits assigning one or more of the columns not accessed by the TPC-H query set to a vertical partition.*

Vertical partitioning was not used. See Appendix B, which contains the database and table creation statements.

2.5 Replication

Any replication of physical objects must be disclosed and must conform to the requirements of Clause1.5.6.

No replication was used. See Appendix B, which contains the database and table creation statements.

3 Clause 2 : QUERIES AND UPDATE FUNCTIONS

3.1 Query Language

The query language used to implement the queries must be identified.

SQL was the query language used to implement all queries.

3.2 Random Number Generation

The method of verification for the random number generation must be described unless the supplied DBGEN and QGEN were used.

DBGEN version 1.3.0 and QGEN version 1.3.0 were used to generate random numbers for these runs.

3.3 Substitution Parameters

The method used to generate values for substitution parameters must be disclosed. If QGEN is not used for this purpose, then the source code of any non-commercial tool used must be disclosed. If QGEN is used, the version number, release number, modification number and patch level of QGEN must be disclosed.

The supplied QGEN version 1.3.0 was used.

3.4 Query Text and Output Data from Qualification Database

The executable query text used for query validation must be disclosed along with the corresponding output data generated during the execution of the query text against the qualification database. If minor modifications (see Clause 2.2.3) have been applied to any functional query definitions or approved variants in order to obtain executable query text, these modifications must be disclosed and justified. The justification for a particular minor query modification can apply collectively to all queries for which it has been used. The output data for the power and throughput tests must be made available electronically upon request.

Appendix C contains the query text and query output. The following allowed minor query modifications were used in this implementation:

- The “dateadd” function is used to perform date arithmetic in Q1, Q4, Q5, Q6, Q10, Q12, Q14 , Q15 and Q20.
- The “datepart” function is used to extract part of a date (“YY”) in Q7, Q8 and Q9.
- The “top” function is used to restrict the number of output rows in Q2, Q3, Q10, Q18 and Q21.
- The “count_big” function is used in place of the “count” function in Q1.

3.5 Query Substitution Parameters and Seeds

All the query substitution parameters used during the performance test must be disclosed in tabular format, along with the seeds used to generate these parameters.

Appendix D contains the seed and query substitution parameters.

3.6 Query Isolation Level

The isolation level used to run the queries must be disclosed. If the isolation level does not map closely to one of the isolation levels defined in Clause 3.4, additional descriptive detail must be provided.

The queries and transactions were run with the isolation level “Level 1.”

3.7 Source Code of Refresh Functions

The details of how the refresh functions were implemented must be disclosed (including source code of any non-commercial program used).

The refresh function is part of the implementation-specific driver code included in Appendix E.

4 Clause 3 : DATABASE SYSTEM PROPERTIES

4.1 Atomicity

The system under test must guarantee that 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.

4.1.1 Completed Transaction

Perform the ACID transaction for a randomly selected set of input data and verify that the appropriate rows have been changed in the ORDER, LINEITEM, and HISTORY tables.

1. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID transaction was performed using the order key from Step 1.
3. The ACID transaction was committed.
4. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for the same order key used in Step 1. It was verified that the appropriate rows had been inserted.

4.1.2 Aborted Transaction

Perform the ACID transaction for a randomly selected set of input data, substituting a ROLLBACK of the transaction for the COMMIT of the transaction. Verify that the appropriate rows have not been changed in the ORDER, LINEITEM, and HISTORY tables.

1. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID transaction was performed using the order key from Step 1. The transaction was stopped prior to the commit.
3. The ACID transaction was ROLLED BACK.
4. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for the same order key used in Step 1. It was verified that the appropriate rows had not been changed.

4.2 Consistency

Consistency is the property of the application that requires any execution of transactions to take the database from one consistent state to another.

4.2.1 Consistency Test

Verify that ORDER and LINEITEM tables are initially consistent, submit the required number of ACID transactions with randomly selected input parameters, and re-verify the consistency of the ORDER and LINEITEM tables.

The consistency of the ORDER and LINEITEM tables was verified based on randomly selected values of the column O_ORDERKEY.

1. ACID queries were executed to verify the initial consistent state of the ORDER and LINEITEM tables.
2. More than 100 ACID transactions were submitted from each of 9 execution streams.
3. ACID queries were re-executed to verify the consistent state of the ORDER and LINEITEM tables after the ACID transaction streams.
4. The consistency of the ORDER and LINEITEM tables was re-verified.

To guarantee arithmetic function portability and consistency of results, the following query was executed to verify the consistency between the ORDER and LINEITEM tables:

```
SELECT DISTINCT(o.O_ORDERKEY),
cast(o.O_TOTALPRICE as decimal(20,3)),
l.L_ORDERKEY,
cast(sum(cast( cast( cast(cast(cast(cast(L_EXTENDEDPRICE as decimal(20,3))*100 as integer) as
```

decimal(20,3)) * (1- cast(L_DISCOUNT as decimal(20,3))) as integer)
* (1 + cast(L_TAX as decimal(20,3))) as integer) as decimal(20,3))/100) as decimal(20,3))

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.

4.3.1 Read-Write Conflict with Commit

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction whenthe read-write transaction is committed.

1. An ACID transaction was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID transaction was suspended prior to COMMIT.
2. An ACID query was started for the same O_KEY used in Step 1. The ACID query did not see the uncommitted changes made by the ACID transaction.
3. The ACID transaction was COMMITTED.
4. The ACID query completed.

4.3.2 Read-Write Conflict with Rollback

Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is rolled back.

1. An ACID transaction was started for a randomly selected O_KEY, L_KEY, and DELA. The ACID transaction was suspended prior to ROLLBACK.
2. An ACID query was started for the same O_KEY used in Step 1. The ACID query did not see the uncommitted changes made by the ACID transaction.
3. The ACID transaction was ROLLED BACK.
4. The ACID query completed.

4.3.3 Write-Write Conflict with Commit

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is committed.

1. An ACID transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID transaction was suspended prior to COMMIT.
2. Another ACID transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to COMMIT and T2 completed.
5. It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE+(DELTA1*(T1.L_EXTENDEDPRICE/T1.L_QUANTITY)).

4.3.4 Write-Write Conflict with Rollback

Demonstrate isolation for the write-write conflict of two update transactions when the first transaction is rolled back.

1. An ACID transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. The ACID transaction was suspended prior to ROLLBACK.
2. Another ACID transaction, T2, was started using the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited
4. T1 was allowed to ROLLED BACK and T2 completed
5. It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE.

4.3.5 Concurrent Progress of Read and Write on Different Tables

Demonstrate the ability of read and write transactions affecting different database tables to make progress concurrently.

1. An ACID transaction, T1, was started for a randomly selected O_KEY, L_KEY, and DELTA. T1 was suspended prior to COMMIT.
2. Another ACID transaction, T2 was started using random values for PS_PARTKEY and PS_SUPPKEY.
3. ACID transaction T2 completed.
4. ACID transaction T1 completed and the appropriate rows in the ORDER, LINEITEM, and HISTORY tables were changed.
5. It was verified that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables have been changed.

4.3.6 Read-Only Query Conflict with Update Transaction

Demonstrate that the continuous submission of arbitrary (read-only) queries against one or more tables of the database does not indefinitely delay update transactions affecting those tables from making progress.

1. An ACID transaction, T1, was started, executing Q1 against the qualification database. The substitution parameter was chosen from the interval [0..2159] so that the query ran for a sufficient length of time.
2. Before T1 completed, an ACID transaction, T2, was started using randomly selected values of O_KEY, L_KEY and DELTA.
3. T2 completed before T1 completed.
4. It was verified that the appropriate rows in the ORDER, LINEITEM and HISTORY tables have been changed.

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 3.5.2

4.4.1 Loss of Data and Log

Guarantee the database and committed updates are preserved across a permanent irrecoverable failure of any single durable medium containing TPC-H database tables or recovery log tables.

The database log was stored RAID-6 protected storages. Loss of data test and loss of log test were combined to one test and demonstrated on the qualification database.

1. 9 streams of ACID transactions were started.
2. After at least 100 transactions had occurred on each stream and the streams of ACID transactions were still running, one of the RAID-6 set of log part disks was removed. The running continued without any interruptions.
3. One of the RAID-0 set of data part disks was removed.
4. Fibre system reported system error resulted in IO error for SQL database.
5. The 9 streams of ACID transactions failed and recorded their numbers of committed transactions in success files.
6. The transaction log was backed up.
7. Two new drives were used to replace the removed log and data disks.
8. The database was restored to their state prior to the ACID transaction streams..
9. The backed up transaction log was applied.
10. The counts in the success files and the HISTORY table count were compared and the counts matched.

4.4.2 Loss of Mirrored write-back cache

Guarantee the database and committed updates are preserved across a permanent irrecoverable failure of any single durable medium containing TPC-H database tables or recovery log tables.

The Fibre Array system used for this benchmark has integrated feature of mirrored write-back cache. When a LUN is configured to enable write-back caching, the data on the cache is automatically mirrored on the RAMs in two controller modules, which are powered and protected from loss of power by the controller BBU. Loss of mirrored write-back cache test was demonstrated on the qualification database.

1. 9 streams of ACID transactions were started.
2. After at least 100 transactions had occurred on each stream and the streams of ACID transactions were still running, a controller module, which manages write-back cache of mirrored drives, was pulled off.
3. Fibre system reported system error resulted in IO error for SQL database.
4. Contents on mirrored cache on another controller were automatically saved to temporal disk area and the Fibre Array house was eventually stopped.
5. The 9 streams of ACID transactions failed and recorded their numbers of committed transactions in success files.
6. The Fibre array was re-powered.
7. Saved data was automatically restored to mirrored cache and flushed to drives.
8. The database ran through its recovery mode.
9. The counts in the success files and the HISTORY table count were compared and the counts matched.

4.4.3 Loss of System

Guarantee the database and committed updates are preserved across an instantaneous interruption (systemcrash/system hang) in processing which requires the system to reboot to recover.

The system crash and memory failure tests were combined.

1. 9 streams of ACID transactions were started.
2. After at least 100 transactions had occurred on each stream and the streams of ACID transactions were still running, the system was powered off.
3. When power was restored the system rebooted and the database was restarted.
4. The database ran through its recovery mode.
5. The counts in the success files and the HISTORY table count were compared and the counts matched.

4.4.4 Loss of Memory

Guarantee the database and committed updates are preserved across failure of all or part of memory (loss of contents).

The system crash and memory failure tests were combined. See the previous section.

5 Clause 4 : Scaling and Database Population

5.1 Cardinality of Tables

The cardinality (e.g., the number of rows) of each table of the test database, as it existed at the completion of the database load (see Clause 4.2.5), must be disclosed.

Table	# of ROWS
Orders	4,500,000,000
Lineitem	18,000,048,306
Customer	450,000,000
Parts	600,000,000
Supplier	30,000,000
Partsupp	2,400,000,000
Nation	25
Region	5

5.2 Distribution of Tables and Logs Across Media

The distribution of tables and logs across all media must be explicitly described using a format similar to that shown in the following example for both the tested and priced systems.

Refer to Appendix G for details of the database layout.

5.3 Partitions/Replications Mapping

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

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

Database partitioning and replication were not used.

5.4 Use of RAID

Implementations may use some form of RAID . The RAID level used must be disclosed for each device.

Hardware RAID-6 was used in the implementation for log. Refer to Appendix G for details of RAID configuration.

5.5 DBGEN Modifications

The version number, release number, modification number, and patch level of DBGEN must be disclosed. Any modifications to the DBGEN source code must be disclosed. In the event that a program other than DBGEN was used to populate the database, it must be disclosed in its entirety.

The supplied DBGEN 1.3.0 was used for populating the database.

5.6 Database Load Time

The database load time for the test database (see Clause 4.3) must be disclosed

The Numerical Quantities summary contains the database load time, which was 36:04:45

5.7 Database Storage Ratio

The data storage ratio must be disclosed. It is computed by dividing the total data storage of the priced configuration (expressed inGB) by the size chosen for the test database. The ratio must be reported to the nearest 1/100th, rounded up. For example, a systemconfigured with 96 disks of 2.1 GB capacity for a 100GB test database has a data storage ratio of 2.02.

***Comment**: For the reporting of configured disk capacity, gigabyte (GB) is defined to be 2^30 bytes. Since disk manufacturers typically report disk size using base ten (i.e., GB = 10^9), it may be necessary to convert the advertised size from base ten to base two.*

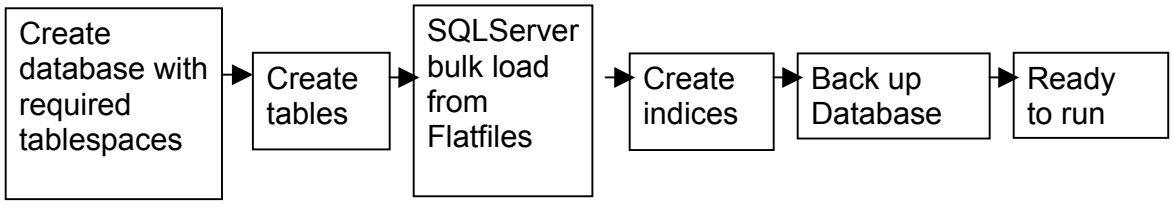
The Numerical Quantities summary contains the data storage ratio (11.66) for the system used.

5.8 Database Loading

The details of the database load must be disclosed, including a block diagram illustrating the overall process. Disclosure of the load procedure includes all steps, scripts, input and configuration files required to completely reproduce the test and qualification databases.

The following steps were used to load the database:

1 DBGEN version 1.3.0 was used to create flat files.
2 SQL Server 2005 was used to define the database, to define tables, and to load the tables via a "bulk insert" command.
3 Clustered indexes were created using SQL Server 2005.
4 Non clustered indexes were created using SQL Server 2005.
5 A database backup was performed to 22 logical devices
6 Rows were inserted into the database by running 96 concurrent threads, each of which performed a "bulk insert" operation that loaded one 96th of each of the LINEITEM, ORDERS, PART, PARTSUPP, SUPPLIER and CUSTOMER tables. The NATION and REGION tables were loaded sequentially, each by a single thread.



5.9 Qualification Database Configuration

Any differences between the configuration of the qualification database and the test database must be disclosed.

The qualification database was created using scripts identical to those of the test database, except for variances due to the sizes of the two databases.

6 Clause 5: Performance Metrics and Execution Rules

6.1 System Activity Between Load and Performance Tests

Any system activity on the SUT which takes place between the conclusion of the load test and the beginning of the performance test must be fully disclosed including listings of scripts or command logs.

Auditor requested queries were run against the database to verify the completeness and correctness of the database load.

6.2 Power Test Implementation

The details of the steps followed to implement the power test (e.g., system boot, database restart, etc.) must be disclosed.

The following steps were followed to run the power test.

1. SQL Server 2005 was started.
2. RF1 refresh transactions were run
3. Stream 00 execution was run
4. RF2 refresh transactions were run.

6.3 Timing Intervals and Reporting

The timing intervals for each query and for both refresh functions must be reported for the power test.

This information is contained in the Numerical Quantities Summary page in the Executive Summary at the beginning of this report. For convenience, it is repeated in Section 6.10.

6.4 Number of Streams in the Throughput Test

The number of query streams used for the throughput test must be disclosed

8 streams were run for the throughput test

6.5 Start and End Date/Time for Each Query Stream

The start time and finish time for each query stream must be reported for the throughput test

This information is contained in the Numerical Quantities Summary page in the Executive Summary at the beginning of this report. For convenience, it is repeated in Section 6.10.

6.6 Total Elapsed Time for the Measurement Interval

The total elapsed time of the measurement interval must be reported for the throughput test.

This information is contained in the Numerical Quantities Summary page in the Executive Summary at the beginning of this report. For convenience, it is repeated in Section 6.10.

6.7 Refresh Function Start Date/Time and Finish Date/Time

The start time and finish time for each refresh function in the refresh stream must be reported for the throughput test.

This information is contained in the Numerical Quantities Summary page in the Executive Summary at the beginning of this report. For convenience, it is repeated in Section 6.10.

6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream

The timing intervals for each query of each stream and for each refresh function must be reported for the throughput test.

This information is contained in the Numerical Quantities Summary page in the Executive Summary at the beginning of this report. For convenience, it is repeated in Section 6.10.

6.9 Performance Metrics

The computed performance metric, related numerical quantities and the price performance metric must be reported.

This information is contained in the Numerical Quantities Summary section of the Executive Summary. For convenience, it is repeated in Section 6.10.

6.10 The Performance Metric and Numerical Quantities from Both Runs

The performance metric (QphH) and the numerical quantities (TPC-H Power@Size and TPC-H Throughput@Size) from both of the runs must be disclosed.

	QppH@3000GB	QthH@3000GB	QphH@3000GB
Run 1	38826.5	23773.5	30381.6
Run 2	34652.8	23259.3	28390.1
% Difference	-12.04%	-2.21%	-7.01%

(Run 2 was reported.)

Tables from Numerical Quantities pages in the Executive Summary of this report:

Numerical Quantities Summary

Measurement Results

Scale Factor	3000 GB
Total Data Storage / Database Size	11.66
Start of Database Load	10/24/2005 12:41:40
End of Database Load	10/26/2005 00:46:25
Database Load Time	36:04:45
Query Streams for Throughput Test	8
TPC-H Power	34652.8
TPC-H Throughput	23259.3
Composite Query per Hour Rating (QphH@3000GB)	28390.1
Total System Price Over 3 Years	$1,888,276
TPC-H Price Performance Metric	$66.52

Measurement Intervals

Measurement Interval in Throughput Test (Ts)	81722 seconds

Duration of Stream Execution:

	Seed	Query Start Date/Time Query End Date/Time	RF1 Start Date/Time RF1 End Date/Time	RF2 Start Date/Time RF2 End Date/Time	Query Duration RF Duration
Stream 0	1026004625	10/27/2005 13:29:41 10/27/2005 16:36:59	10/27/2005 13:03:10 10/27/2005 13:29:40	10/27/2005 16:40:00 10/27/2005 16:46:55	
Stream 1	1026004626	10/27/2005 16:46:58 10/28/2005 11:15:30	10/28/2005 11:34:58 10/28/2005 11:46:16	10/28/2005 11:46:21 10/28/2005 11:56:01	18:28:32 0:20:59
Stream 2	1026004627	10/27/2005 16:46:58 10/28/2005 11:29:36	10/28/2005 11:56:07 10/28/2005 12:09:05	10/28/2005 12:09:08 10/28/2005 12:21:11	18:42:38 0:25:01
Stream 3	1026004628	10/27/2005 16:46:59 10/28/2005 11:22:47	10/28/2005 12:21:20 10/28/2005 12:35:26	10/28/2005 12:35:31 10/28/2005 12:47:50	18:35:48 0:26:25
Stream 4	1026004629	10/27/2005 16:47:00 10/28/2005 11:27:56	10/28/2005 12:48:01 10/28/2005 13:02:48	10/28/2005 13:02:54 10/28/2005 13:15:44	18:40:55 0:27:36
Stream 5	1026004630	10/27/2005 16:47:01 10/28/2005 10:05:04	10/28/2005 13:15:58 10/28/2005 13:31:54	10/28/2005 13:32:00 10/28/2005 13:45:15	17:18:03 0:29:11
Stream 6	1026004631	10/27/2005 16:47:03 10/28/2005 11:10:12	10/28/2005 13:45:39 10/28/2005 14:02:42	10/28/2005 14:02:49 10/28/2005 14:17:14	18:23:09 0:31:27
Stream 7	1026004632	10/27/2005 16:47:05 10/28/2005 11:30:38	10/28/2005 14:17:57 10/28/2005 14:36:23	10/28/2005 14:36:48 10/28/2005 14:51:53	18:43:33 0:33:31
Stream 8	1026004633	10/27/2005 16:47:07 10/28/2005 11:34:58	10/28/2005 14:52:41 10/28/2005 15:12:58	10/28/2005 15:13:32 10/28/2005 15:29:00	18:47:52 0:35:44

TPC-H Timing Intervals (in seconds):

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream 0	1164.2	83.8	80.0	129.6	259.7	106.2	295.3	354.8
Stream 1	6453.1	469.9	846.3	5324.2	2896.3	278.0	2051.2	1213.9
Stream 2	7918.3	1338.6	1199.0	1301.7	2890.0	560.3	2804.6	871.1
Stream 3	7127.4	211.3	4332.9	719.7	5635.7	281.5	2984.4	991.8
Stream 4	7821.6	306.7	3626.6	1360.9	867.7	193.0	1054.7	2468.1
Stream 5	9609.0	2361.3	2292.8	887.4	1781.6	347.4	4188.8	3269.1
Stream 6	7766.5	1013.4	5247.6	3938.5	2570.1	392.9	1950.9	1977.2
Stream 7	7419.5	1935.6	2965.3	3146.7	1020.2	330.3	2025.1	1379.6
Stream 8	10395.0	843.9	1535.3	1170.6	3132.0	284.9	2742.5	2508.8
Min Qi	6453.1	211.3	846.3	719.7	867.7	193.0	1054.7	871.1
Max Qi	10395.0	2361.3	5247.6	5324.2	5635.7	560.3	4188.8	3269.1
Avg Qi	8063.8	1060.1	2755.7	2231.2	2599.2	333.5	2475.3	1835.0
Query	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
Stream 0	1832.6	141.3	236.3	373.7	445.5	109.8	58.2	933.6
Stream 1	11824.8	1552.7	2697.0	1922.3	3708.8	867.5	1646.1	2614.4
Stream 2	9556.6	1701.9	1012.6	2767.6	2606.7	4574.4	577.8	2578.9
Stream 3	5279.1	1567.9	1917.7	3769.7	3037.5	2013.7	345.3	3846.7
Stream 4	4881.8	1744.6	1800.2	3961.8	2183.7	1232.8	1222.3	3171.3
Stream 5	4209.3	952.9	1914.2	3160.5	3104.8	686.2	754.3	3478.4
Stream 6	5077.0	1340.2	1318.3	2551.1	4019.8	1224.7	511.5	3289.6
Stream 7	7971.5	1290.3	1041.6	614.7	2801.1	5224.7	169.4	3311.3
Stream 8	7391.7	1304.7	215.8	3042.3	4015.3	2347.5	1078.8	4012.4
Min Qi	4209.3	952.9	215.8	614.7	2183.7	686.2	169.4	2578.9
Max Qi	11824.8	1744.6	2697.0	3961.8	4019.8	5224.7	1646.1	4012.4
Avg Qi	7024.0	1431.9	1489.7	2723.8	3184.7	2271.4	788.2	3287.9
Query	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 0	163.6	1956.0	312.2	148.2	2032.9	200.0	1590.7	414.7
Stream 1	1081.2	6771.5	1357.3	3629.8	6085.8	1219.6	678.1	580.3
Stream 2	1712.6	6538.0	2925.5	1556.4	8008.7	2356.5	778.2	722.8
Stream 3	2330.0	6666.0	1551.5	543.2	9373.6	2421.0	846.7	738.6
Stream 4	1339.7	6328.3	3455.4	4781.0	12076.4	1377.0	886.2	769.7
Stream 5	1867.7	7542.2	1893.6	524.2	6080.6	1376.1	956.2	794.4
Stream 6	1171.4	7371.8	2025.8	2344.0	7627.6	1459.0	1023.2	864.2
Stream 7	538.7	6367.3	4380.3	2298.4	9645.8	1535.8	1106.0	905.0
Stream 8	2353.9	6714.3	2451.7	686.8	7108.0	2335.5	1216.8	927.7
Min Qi	538.7	6328.3	1357.3	524.2	6080.6	1219.6	678.1	580.3
Max Qi	2353.9	7542.2	4380.3	4781.0	12076.4	2421.0	1216.8	927.7
Avg Qi	1549.4	6787.4	2505.1	2045.5	8250.8	1760.1	936.4	787.8

6.11 System Activity Between Tests

Any activity on the SUT that takes place between the conclusion of Run1 and the beginning of Run2 must be fully disclosed including listings of scripts or command logs along with any system reboots or database restarts.

The following activities took place between the conclusion of Run 1 and the beginning of Run 2:

1. Shutdown SQL Server.
2. Rebooted System.
3. Restarted SQL Server.

7 Clause 6 : SUT and Driver Implementation Related Items

7.1 Driver

A detailed textual description of how the driver performs its functions, how its various components interact and any product functionalities or environmental setting on which it relies must be provided. All related source code, scripts and configuration files must be disclosed. The information provided should be sufficient for an independent reconstruction of the driver.

The TPC-H benchmark was implemented using a Microsoft internal tool called StepMaster. StepMaster is a general purpose test harness which can drive ODBC and shell commands. Within StepMaster, the user designs a workspace corresponding to the sequence of operations (or steps) to be executed. When the workspace is executed, StepMaster records information about the run into a database for post-processing.

StepMaster provides a mechanism for creating parallel streams of execution. This is used in the throughput tests to drive the query and refresh streams.

Each step is timed using a millisecond resolution timer. A timestamp T1 is taken before beginning the operation and a timestamp T2 is taken after completing the operation. These times are recorded in a database for post-processing.

Two types of ODBC connections are supported: static and dynamic. A dynamic connection is used to execute a single operation and is closed when the operation finishes. A static connection is held open until the run completes and may be used to execute more than one step. A connection (either static or dynamic) can only have one outstanding operation at any time.

In TPC-H, static connections are used for the query streams in the power and throughput tests.

StepMaster reads an Access database to determine the sequence of steps to execute. These commands are represented as the Implementation Specific Layer. StepMaster records its execution history, including all timings, in the Access database. Additionally, StepMaster writes a textual log file of execution for each run.

SQL Server operations executed from StepMaster do not gain any performance advantage compared to osql, the command prompt utility for ad hoc, interactive execution of Transact-SQL statements and scripts. Rather, StepMaster simplifies the task of benchmark execution, event timing, and reporting. This was confirmed during the audit.

7.2 Implementation-Specific Layer (ISL)

If an implementation specific layer is used, then a detailed description of how it performs its functions, how its various components interact and any product functionalities or environmental setting on which it relies must be provided. All related source code, scripts and configuration files must be disclosed. The information provided should be sufficient for an independent reconstruction of the implementation specific layer.

StepMaster program is used to control and track the execution of queries, via commands stored in an external Microsoft Access database. The source of this program is contained in Appendix F. The following steps are performed, to accomplish the Power and Throughput Runs:

1 **Power Run**

- Execute 64 concurrent RF1 threads, each of which will apply a segment of a refresh set generated by dbgen. Each thread submits multiple transactions, where a transaction spans a set of orders and their associated line items.
- Execute the Stream 0 queries, in the prescribed order.
- Execute 64 concurrent RF2 threads, each of which will apply a segment of a refresh set generated by dbgen. Each thread submits multiple transactions, where a transaction spans a set of orders and their associated line items.

2 Throughput Run

- Execute 8 concurrent query streams. Each stream executes queries in the prescribed order for the appropriate Stream Id (01-08). Upon completion of each stream, a semaphore is set to indication completion.
- Execute 8 consecutive RF1/RF2 transactions, against ascending Refresh sets produced by dbgen. The first RF1 waits on a semaphore prior to beginning its insert operations.

Each step is timed by StepMaster. The timing information, together with an activity log, are stored for later analysis. The inputs and results of steps are stored in text files for later analysis.

8 Clause 7 : Pricing Related Items

8.1 Hardware and Software Used

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

The detailed list of all hardware and software for the priced configuration is listed in the system pricing summary.

8.2 Three-Year Cost of System Configuration

The total 3-year price of the entire configuration must be reported, including: hardware, software, and maintenance charges. Separate component pricing is required.

The detailed list of 3-years maintenance cost for the priced configuration is listed in the system pricing summary.

8.3 Availability Dates

The committed delivery date for general availability (availability date) of products used in the priced calculations must be reported. When the priced system includes products with different availability dates, the single availability date reported on the first page of the executive summary must be the date by which all components are committed to being available. The full disclosure report must report availability dates individually for at least each of the categories for which a pricing subtotal must be provided (see Clause 7.3.1.4). All availability dates, whether for individual components or for the SUT as a whole, must be disclosed to a precision of 1 day, but the precise format is left to the test sponsor.

The total system as priced will be available by May 5, 2006.

- "Microsoft SQL Server 2005 Enterprise Itanium Edition" will be available by May 5, 2006.
- The other products are already available.

9 Clause 8 : Audit Related Items

The auditor's agency name, address, phone number, and Attestation letter with a brief audit summary report indicating compliance must be included in the full disclosure report. A statement should be included specifying who to contact in order to obtain further information regarding the audit process.

See pages in the front of this paper for a copy of the auditor's attestation letter. Further information regarding the audit process may be obtained from Mr. Raab.

Appendix A : System and Database Tunable Parameters

Software Level

Microsoft Windows Server 2003,Datacenter Edition
for 64-bit Itanium-based Systems, Service Pack 1
Microsoft SQL Server 2005 Enterprise Itanium Edition build 1399

User Rights Assignment

The Group Policy Editor was used to modify an entry under "Computer Configuration" -> "Windows Settings" -> "Security Settings" -> "Local Policies" -> "User Rights Assignment". The right of "Lock pages in memory" was assigned to the Administrator so that SQL Server could use large amounts of physical memory.

System Information

```
System Information report written at: 10/29/05 07:27:33
System Name: ASAMA
[System Summary]

Item    Value
OS Name Microsoft(R) Windows(R) Server 2003, Datacenter Edition for 64-
Bit Itanium-based Systems
Version 5.2.3790 Service Pack 1 Build 3790
Other OS Description    Not Available
OS Manufacturer         Microsoft Corporation
System Name     ASAMA
System Manufacturer     NEC
System Model    server u32000
System Type     Itanium (TM) -based System
Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

Processor       ia64 Family 31 Model 2 Stepping 2 GenuineIntel ~1600 Mhz

BIOS Version/Date       Phoenix Technologies LTD 6.0.0604, 3/2/2005
SMBIOS Version 2.3
Windows Directory       C:\WINDOWS
System Directory        C:\WINDOWS\system32
Boot Device     \Device\HarddiskVolume18
Locale  United States
Hardware Abstraction Layer      Version = "5.2.3790.1830
(srv03_sp1_rtm.050324-1447)"
User Name       ASAMA\Administrator
Time Zone       Tokyo Standard Time
Total Physical Memory   262,112.14 MB
Available Physical Memory       7.70 GB
Total Virtual Memory    255.59 GB
Available Virtual Memory        8.43 GB
Page File Space         2.00 GB
```

```
Page File      C:\pagefile.sys
```

Microsoft® SQL Server™ 2005 Startup Parameters

```
sqlservr -c -x -E -T834 -T2301 -T3502

-c
  Start SQL Server independently of the Windows Service Control Manager

-x
  Disable the keeping of CPU time and cache-hit ration statistics.

-E
  increase the number of consecutive extents allocated per file to 4

-T834
  On systems with 8GB or more, this traceflag causes the buffer pool to
  use large pages. These are allocated at startup and are kept throughout
  the lifetime of the process. This trace flag can only be set on 64-bit
  installations of SQL Server.

-T2301
  trace flag to enable more accurate query run-time behavior modeling
  in the SQL Server query optimizer typically only needed for large
  data set decision support processing.

-T3502
  Prints a message to the log at the beginning and end of each checkpoint.
```

Microsoft® SQL Server™ 2005 Configuration Parameters

name	minimum	maximum	config_value	run_value
Ad Hoc Distributed Queries	0	1	0	0
affinity I/O mask	-2147483648	2147483647	0	0
affinity mask	-2147483648	2147483647	-1	-1
affinity64 I/O mask	-2147483648	2147483647	0	0
affinity64 mask	-2147483648	2147483647	0	0
Agent XPs	0	1	0	0
allow updates	0	1	1	1
awe enabled	0	1	0	0
blocked process threshold	0	86400	0	0
c2 audit mode	0	1	0	0
clr enabled	0	1	0	0
cost threshold for parallelism	0	32767	0	0
cross db ownership chaining	0	1	0	0
cursor threshold	-1	2147483647	-1	-1
Database Mail XPs	0	1	0	0
default full-text language	0	2147483647	1033	1033
default language	0	9999	0	0
default trace enabled	0	1	1	1
disallow results from triggers	0	1	0	0
fill factor (%)	0	100	0	0
ft crawl bandwidth (max)	0	32767	100	100
ft crawl bandwidth (min)	0	32767	0	0
ft notify bandwidth (max)	0	32767	100	100
ft notify bandwidth (min)	0	32767	0	0
in-doubt xact resolution	0	2	0	0
index create memory (KB)	704	2147483647	0	0
lightweight pooling	0	1	1	1
locks	5000	2147483647	0	0
max degree of parallelism	0	64	0	0
max full-text crawl range	0	256	4	4
max server memory (MB)	16	2147483647	250000	250000
max text repl size (B)	0	2147483647	65536	65536
max worker threads	128	32767	2000	2000
media retention	0	365	0	0
min memory per query (KB)	512	2147483647	512	512
min server memory (MB)	0	2147483647	232000	232000
nested triggers	0	1	1	1
network packet size (B)	512	32767	4096	4096
Ole Automation Procedures	0	1	0	0
open objects	0	2147483647	0	0
PH timeout (s)	1	3600	60	60
precompute rank	0	1	0	0
priority boost	0	1	0	0

```
                0
query governor cost limit                   0  2147483647           0
                0
query wait (s)                             -1  2147483647  2147483647
       2147483647
recovery interval (min)                     0       32767       32767
            32767
remote access                               0           1           1
                1
remote admin connections                    0           1           0
                0
remote login timeout (s)                    0  2147483647          20
               20
remote proc trans                           0           1           0
                0
remote query timeout (s)                    0  2147483647           0
                0
Replication XPs                             0           1           0
                0
scan for startup procs                      0           1           0
                0
server trigger recursion                    0           1           1
                1
set working set size                        0           1           0
                0
show advanced options                       0           1           1
                1
SMO and DMO XPs                             0           1           1
                1
SQL Mail XPs                                0           1           0
                0
transform noise words                       0           1           0
                0
two digit year cutoff                    1753        9999        2049
             2049
user connections                            0       32767           0
                0
user options                                0       32767           0
                0
Web Assistant Procedures                    0           1           0
                0
xp_cmdshell                                 0           1           0
                0
```

Appendix B : Database, table and indexes creation

CreateClusteredIndexes1.sql

```
--  File:     CREATECLUSTEREDINDEXES.SQL
--            Microsoft TPC-H Benchmark Kit Ver. 1.00
--            Copyright Microsoft, 1999
--

 create clustered index L_SHIPDATE_CLUIDX
       on LINEITEM(L_SHIPDATE)
       with FILLFACTOR=95,SORT_IN_TEMPDB
       on LINEITEM_FG
```

CreateClusteredIndexes2.sql

```
--  File:     CREATECLUSTEREDINDEXES.SQL
--            Microsoft TPC-H Benchmark Kit Ver. 1.00
--            Copyright Microsoft, 1999
--

 create clustered index O_ORDERDATE_CLUIDX
       on ORDERS(O_ORDERDATE)
       with FILLFACTOR=95,SORT_IN_TEMPDB
       on GENERAL_FG

 alter table NATION add constraint PK_N_NATIONKEY primary key
(N_NATIONKEY)
 ON GENERAL_FG

 alter table REGION add constraint PK_R_REGIONKEY primary key
(R_REGIONKEY)
 ON GENERAL_FG

 alter table PART add constraint PK_P_PARTKEY primary key (P_PARTKEY)
 ON GENERAL_FG

 alter table SUPPLIER add constraint PK_S_SUPPKEY primary key
(S_SUPPKEY)
 ON GENERAL_FG

 alter table CUSTOMER add constraint PK_C_CUSTKEY primary key
(C_CUSTKEY)
 ON GENERAL_FG

 alter table PARTSUPP add constraint PK_PS_PARTKEY_PS_SUPPKEY primary
key (PS_PARTKEY, PS_SUPPKEY)
 ON GENERAL_FG

 alter table ORDERS add constraint PK_O_ORDERKEY primary key
(O_ORDERKEY)
 WITH (FILLFACTOR = 95)
 ON GENERAL_FG
```

CreateDatabase.sql

```
--  CreateDatabase
--  for use with StepMaster
--  Uses FileGroups

--
--      Drop the existing database
--

if exists (select name from sysdatabases where name = '%DBNAME%')
       drop database %DBNAME%

CREATE DATABASE %DBNAME%
ON PRIMARY
(      NAME           = %DBNAME%_root,
```

```
       FILENAME       = "z:\%DBNAME%_root.mdf",
       SIZE           = 10MB,
       FILEGROWTH     = 10MB),
FILEGROUP      LINEITEM_FG
(NAME=%DBNAME%_li_01, FILENAME="z:\dev\c001\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_02, FILENAME="z:\dev\c002\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_03, FILENAME="z:\dev\c003\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_04, FILENAME="z:\dev\c004\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_05, FILENAME="z:\dev\c005\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_06, FILENAME="z:\dev\c006\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_07, FILENAME="z:\dev\c007\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_08, FILENAME="z:\dev\c008\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_09, FILENAME="z:\dev\c009\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_10, FILENAME="z:\dev\c010\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_11, FILENAME="z:\dev\c011\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_12, FILENAME="z:\dev\c012\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_13, FILENAME="z:\dev\c013\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_14, FILENAME="z:\dev\c014\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_15, FILENAME="z:\dev\c015\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_16, FILENAME="z:\dev\c016\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_17, FILENAME="z:\dev\c017\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_18, FILENAME="z:\dev\c018\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_19, FILENAME="z:\dev\c019\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_20, FILENAME="z:\dev\c020\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_21, FILENAME="z:\dev\c021\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_22, FILENAME="z:\dev\c022\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_23, FILENAME="z:\dev\c023\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_24, FILENAME="z:\dev\c024\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_25, FILENAME="z:\dev\c025\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_26, FILENAME="z:\dev\c026\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_27, FILENAME="z:\dev\c027\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_28, FILENAME="z:\dev\c028\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_29, FILENAME="z:\dev\c029\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_30, FILENAME="z:\dev\c030\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_31, FILENAME="z:\dev\c031\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_32, FILENAME="z:\dev\c032\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_33, FILENAME="z:\dev\c033\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_34, FILENAME="z:\dev\c034\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_35, FILENAME="z:\dev\c035\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_36, FILENAME="z:\dev\c036\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_37, FILENAME="z:\dev\c037\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_38, FILENAME="z:\dev\c038\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_39, FILENAME="z:\dev\c039\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_40, FILENAME="z:\dev\c040\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_41, FILENAME="z:\dev\c041\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_42, FILENAME="z:\dev\c042\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_43, FILENAME="z:\dev\c043\", SIZE=240000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_li_44, FILENAME="z:\dev\c044\", SIZE=240000MB,
FILEGROWTH=0),
FILEGROUP      GENERAL_FG
(NAME=%DBNAME%_gen_01, FILENAME="z:\dev\m001\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_02, FILENAME="z:\dev\m002\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_03, FILENAME="z:\dev\m003\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_04, FILENAME="z:\dev\m004\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_05, FILENAME="z:\dev\m005\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_06, FILENAME="z:\dev\m006\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_07, FILENAME="z:\dev\m007\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_08, FILENAME="z:\dev\m008\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_09, FILENAME="z:\dev\m009\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_10, FILENAME="z:\dev\m010\", SIZE=120000MB,
```

```
FILEGROWTH=0),
(NAME=%DBNAME%_gen_11, FILENAME="z:\dev\m011\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_12, FILENAME="z:\dev\m012\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_13, FILENAME="z:\dev\m013\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_14, FILENAME="z:\dev\m014\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_15, FILENAME="z:\dev\m015\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_16, FILENAME="z:\dev\m016\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_17, FILENAME="z:\dev\m017\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_18, FILENAME="z:\dev\m018\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_19, FILENAME="z:\dev\m019\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_20, FILENAME="z:\dev\m020\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_21, FILENAME="z:\dev\m021\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_22, FILENAME="z:\dev\m022\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_23, FILENAME="z:\dev\m023\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_24, FILENAME="z:\dev\m024\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_25, FILENAME="z:\dev\m025\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_26, FILENAME="z:\dev\m026\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_27, FILENAME="z:\dev\m027\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_28, FILENAME="z:\dev\m028\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_29, FILENAME="z:\dev\m029\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_30, FILENAME="z:\dev\m030\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_31, FILENAME="z:\dev\m031\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_32, FILENAME="z:\dev\m032\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_33, FILENAME="z:\dev\m033\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_34, FILENAME="z:\dev\m034\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_35, FILENAME="z:\dev\m035\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_36, FILENAME="z:\dev\m036\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_37, FILENAME="z:\dev\m037\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_38, FILENAME="z:\dev\m038\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_39, FILENAME="z:\dev\m039\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_40, FILENAME="z:\dev\m040\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_41, FILENAME="z:\dev\m041\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_42, FILENAME="z:\dev\m042\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_43, FILENAME="z:\dev\m043\", SIZE=120000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_gen_44, FILENAME="z:\dev\m044\", SIZE=120000MB,
FILEGROWTH=0)
LOG ON
(NAME=%DBNAME%_log_01, FILENAME="z:\dev\log1\", SIZE=380000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_log_02, FILENAME="z:\dev\log2\", SIZE=380000MB,
FILEGROWTH=0),
(NAME=%DBNAME%_log_03, FILENAME="z:\dev\log3\", SIZE=380000MB,
FILEGROWTH=0)
```

createFK.sql

```
-- create FK
alter table SUPPLIER add constraint FK_S_NATIONKEY foreign key
(S_NATIONKEY) references NATION(N_NATIONKEY)

alter table PARTSUPP add constraint FK_PS_PARTKEY foreign key
(PS_PARTKEY) references PART(P_PARTKEY)

alter table PARTSUPP add constraint FK_PS_SUPPKEY foreign key
(PS_SUPPKEY) references SUPPLIER(S_SUPPKEY)

alter table CUSTOMER add constraint FK_C_NATIONKEY foreign key
(C_NATIONKEY) references NATION (N_NATIONKEY)

alter table ORDERS add constraint FK_O_CUSTKEY foreign key (O_CUSTKEY)
references CUSTOMER (C_CUSTKEY)

alter table NATION add constraint FK_N_REGIONKEY foreign key
(N_REGIONKEY)  references REGION (R_REGIONKEY)

alter table LINEITEM add constraint FK_L_ORDERKEY foreign key
(L_ORDERKEY)  references ORDERS (O_ORDERKEY)

alter table LINEITEM add constraint FK_L_PARTKEY foreign key (L_PARTKEY)
references PART (P_PARTKEY)

alter table LINEITEM add constraint FK_L_SUPPKEY foreign key (L_SUPPKEY)
references SUPPLIER (S_SUPPKEY)

alter table LINEITEM add constraint FK_L_PARTKEY_SUPPKEY foreign key
(L_PARTKEY,L_SUPPKEY) references PARTSUPP(PS_PARTKEY, PS_SUPPKEY)
```

createindexesstream1.sql

```
--  File:     CREATEINDEXESSTREAM2.SQL
--            Microsoft TPC-H Benchmark Kit Ver. 1.00
--            Copyright Microsoft, 1999
--

 create index O_CUSTKEY_IDX
       on ORDERS(O_CUSTKEY)
       with fillfactor=95, SORT_IN_TEMPDB
       on GENERAL_FG

 create index PS_SUPPKEY_IDX
        on PARTSUPP(PS_SUPPKEY)
        with fillfactor=100, SORT_IN_TEMPDB
        on GENERAL_FG

 create index N_REGIONKEY_IDX
        on NATION(N_REGIONKEY)
        with fillfactor=100, SORT_IN_TEMPDB
        on GENERAL_FG

 create index S_NATIONKEY_IDX
       on SUPPLIER(S_NATIONKEY)
       with fillfactor=100, SORT_IN_TEMPDB
       on GENERAL_FG
```

createindexesstream2.sql

```
--  File:     CREATEINDEXESSTREAM3.SQL
--            Microsoft TPC-H Benchmark Kit Ver. 1.00
--            Copyright Microsoft, 1999
--

 create index L_ORDERKEY_IDX
       on LINEITEM(L_ORDERKEY)
       with FILLFACTOR=95,SORT_IN_TEMPDB
       on LINEITEM_FG
```

createindexesstream3.sql

```
--  File:     CREATEINDEXESSTREAM4.SQL
--            Microsoft TPC-H Benchmark Kit Ver. 1.00
--            Copyright Microsoft, 1999
--

 create index L_PARTKEY_IDX
       on LINEITEM(L_PARTKEY)
       with FILLFACTOR=95,SORT_IN_TEMPDB
       on LINEITEM_FG
```

CreateTables.sql

```
--  File:     CREATETABLES.SQL
--            Microsoft TPC-H Benchmark Kit Ver. 1.00
--            Copyright Microsoft, 1999
--

create table PART
       (P_PARTKEY     int            not null,
        P_NAME        varchar(55)    not null,
        P_MFGR        char(25)       not null,
        P_BRAND       char(10)       not null,
        P_TYPE        varchar(25)    not null,
        P_SIZE        int            not null,
        P_CONTAINER   char(10)       not null,
        P_RETAILPRICE float          not null,
        P_COMMENT     varchar(23)    not null)
on GENERAL_FG

create table SUPPLIER
       (S_SUPPKEY     int            not null,
        S_NAME        char(25)       not null,
        S_ADDRESS     varchar(40)    not null,
        S_NATIONKEY   int            not null,
        S_PHONE       char(15)       not null,
        S_ACCTBAL     float          not null,
        S_COMMENT     varchar(101)   not null)
on GENERAL_FG

create table PARTSUPP
       (PS_PARTKEY    int            not null,
        PS_SUPPKEY    int            not null,
        PS_AVAILQTY   int            not null,
        PS_SUPPLYCOST float          not null,
        PS_COMMENT    varchar(199)   not null)
on GENERAL_FG

create table CUSTOMER
       (C_CUSTKEY     int            not null,
        C_NAME        varchar(25)    not null,
        C_ADDRESS     varchar(40)    not null,
        C_NATIONKEY   int            not null,
        C_PHONE       char(15)       not null,
        C_ACCTBAL     float          not null,
        C_MKTSEGMENT  char(10)       not null,
        C_COMMENT     varchar(117)   not null)
on GENERAL_FG

create table ORDERS
       (O_ORDERKEY    bigint         not null,
        O_CUSTKEY     int            not null,
        O_ORDERSTATUS char(1)        not null,
        O_TOTALPRICE  float          not null,
        O_ORDERDATE   datetime       not null,
        O_ORDERPRIORITY char(15)     not null,
        O_CLERK       char(15)       not null,
        O_SHIPPRIORITY       int           not null,
        O_COMMENT     varchar(79)    not null)
on GENERAL_FG
```

```
create table LINEITEM
        (L_ORDERKEY    bigint         not null,
         L_PARTKEY     int            not null,
         L_SUPPKEY     int            not null,
         L_LINENUMBER  int            not null,
         L_QUANTITY    float          not null,
         L_EXTENDEDPRICE float        not null,
         L_DISCOUNT    float          not null,
         L_TAX         float          not null,
         L_RETURNFLAG  char(1)        not null,
         L_LINESTATUS  char(1)        not null,
         L_SHIPDATE    datetime       not null,
         L_COMMITDATE  datetime       not null,
         L_RECEIPTDATE datetime       not null,
         L_SHIPINSTRUCT       char(25)       not null,
         L_SHIPMODE    char(10)       not null,
         L_COMMENT     varchar(44)    not null)
on LINEITEM_FG

create table NATION
        (N_NATIONKEY   int            not null,
         N_NAME        char(25)       not null,
         N_REGIONKEY   int            not null,
         N_COMMENT     varchar(152)   not null)
on GENERAL_FG

create table REGION
        (R_REGIONKEY   int            not null,
         R_NAME        char(25)       not null,
         R_COMMENT     varchar(152)   not null)
on GENERAL_FG
```

Appendix C : Query text & output

EXECUTE STREAM 0 QUERY 1

```
-------------------------------------------------------------

-- using 1026004625 as a seed to the RNG


/* TPC_H  Query 1 - Pricing Summary Report */

SELECT  L_RETURNFLAG,
        L_LINESTATUS,
        SUM(L_QUANTITY)                                         AS SUM_QTY,
        SUM(L_EXTENDEDPRICE)                            AS SUM_BASE_PRICE,
        SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT))             AS SUM_DISC_PRICE,
        SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1+L_TAX))         AS
SUM_CHARGE,
        AVG(L_QUANTITY)                                         AS AVG_QTY,
        AVG(L_EXTENDEDPRICE)                            AS AVG_PRICE,
        AVG(L_DISCOUNT)                                         AS AVG_DISC,
        COUNT_BIG(*)                                            AS COUNT_ORDER
FROM    LINEITEM
WHERE   L_SHIPDATE     <= dateadd(dd, -120, '1998-12-01')
GROUP   BY      L_RETURNFLAG,
                L_LINESTATUS
ORDER   BY      L_RETURNFLAG,
                L_LINESTATUS



OUTPUT OF EXECUTE STREAM 0 QUERY 1 (MAX 200 ROWS)
-------------------------------------------------------------

L_RETURNFLAG L_LINESTATUS SUM_QTY                  SUM_BASE_PRICE
SUM_DISC_PRICE           SUM_CHARGE               AVG_QTY
AVG_PRICE                AVG_DISC                 COUNT_ORDER
------------ ------------ ------------------------ --------------------
--- ------------------------ ------------------------ ------------------
------ ------------------------ ------------------------ ---------------
-----
A            F            113383938885.000000
170018153216832.530000   161517034929306.370000   167977648601678.840000
25.499814                38236.732415             0.050001
4446461360
N            F            2959911484.000000        4438231735157.829100
4216311756463.419400     4384977884956.021000     25.500387
38236.490192             0.050005                 116073199
N            O            218274480889.000000
327302140441994.250000   310937093867506.870000   323374702499542.310000
25.500106                38237.357852             0.049999
8559747818
R            F            113383359689.000000
170018012228622.840000   161517146341999.250000   167977776008357.220000
25.499831                38236.921557             0.050000
4446435678

(4 row(s) affected)
```

EXECUTE STREAM 0 QUERY 2

```
-------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 2 - Minimum Cost Supplier */

SELECT  TOP 100
        S_ACCTBAL,
        S_NAME,
        N_NAME,
        P_PARTKEY,
        P_MFGR,
        S_ADDRESS,
```

```
        S_PHONE,
        S_COMMENT
FROM    PART,
        SUPPLIER,
        PARTSUPP,
        NATION,
        REGION
WHERE   P_PARTKEY      = PS_PARTKEY AND
        S_SUPPKEY      = PS_SUPPKEY AND
        P_SIZE         = 13 AND
        P_TYPE         LIKE '%NICKEL' AND
        S_NATIONKEY    = N_NATIONKEY AND
        N_REGIONKEY    = R_REGIONKEY AND
        R_NAME         = 'EUROPE' AND
        PS_SUPPLYCOST  = (     SELECT  MIN(PS_SUPPLYCOST)
                               FROM    PARTSUPP,
                                       SUPPLIER,
                                       NATION,
                                       REGION
                               WHERE   P_PARTKEY      = PS_PARTKEY AND
                                       S_SUPPKEY      = PS_SUPPKEY AND
                                       S_NATIONKEY    = N_NATIONKEY AND
                                       N_REGIONKEY    = R_REGIONKEY AND
                                       R_NAME         = 'EUROPE'
                         )
ORDER   BY      S_ACCTBAL DESC,
                N_NAME,
                S_NAME,
                P_PARTKEY
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 2 (MAX 200 ROWS)
-------------------------------------------------------------

S_ACCTBAL               S_NAME                    N_NAME
P_PARTKEY   P_MFGR                    S_ADDRESS
S_PHONE         S_COMMENT
----------------------- ------------------------- ----------------------
---- ----------- ------------------------- ----------------------------
----------- --------------- ---------------------------------------------
-------------------------------------------------------------
9999.990000             Supplier#010929199        RUSSIA
573429179   Manufacturer#2            BgyBXUwMJGSGM
32-680-187-9147 slyly regular pinto beans haggle fluffily unusual
platelets. furiously silen
9999.980000             Supplier#017909171        ROMANIA
392909144   Manufacturer#5            ,uOERevdfsbc0vuEkUFvFoqOUUL0mH7t
y8vCm   29-547-349-6668 furiously regular requests after the unusual, s
9999.980000             Supplier#027938621        UNITED KINGDOM
312938600   Manufacturer#2
4mcF,LOQfxS,AW6sfzSb7nE7El0OZmFRp45      33-172-680-3540 deposits haggle.
special, permanent theodolites against the even platelets
9999.950000             Supplier#018972094        ROMANIA
236472072   Manufacturer#5            TOdf5hWIOuhEAxvKHR
IR4toExiyWktXDEuYr    29-727-665-9085 blithely express instructions
integrate bli
9999.950000             Supplier#025574372        ROMANIA
325574371   Manufacturer#3            PeWLSODJihh9qld7 O4M
29-984-212-5407 slyly even foxes according to the fluffi
9999.940000             Supplier#007193698        FRANCE
224693676   Manufacturer#1            Gh35C50lvvnu64s
16-136-912-5450 pinto beans about the orbits wake blithely carefully
even
9999.940000             Supplier#002897915        ROMANIA
122897914   Manufacturer#2
Wh6O,zscR3WKqZci6tyrSip7QcjC3Eg5v5P1hpAc 29-361-770-5744 ideas detect.
quickly final Tiresias nag blithely final accounts. furiously
9999.940000             Supplier#029218317        ROMANIA
419218316   Manufacturer#1            wQOSaxU d6gfTknkvGX
29-508-954-2624 slyly even notornis are bold mult
9999.940000             Supplier#007462187        UNITED KINGDOM
262462170   Manufacturer#4
A8m49zTygkBlJT4UPBgciS74tjg5IAESUZx7K,   33-640-170-8936 express
sentiments are furiously. accounts are fluffily under the qu
9999.930000             Supplier#015312855        FRANCE
30312852    Manufacturer#3            KLkgqdU8XPfJA3tttfC
16-335-371-3463 ironic, final foxes cajole never regular deposits.
carefully bold deposit
9999.930000             Supplier#009218220        UNITED KINGDOM
166718204   Manufacturer#1            qF  7NqTnlJAYwN4PUdJZxm95K
33-949-902-3942 warthogs haggle. ironic instructions according to the
furiously even warh
9999.930000             Supplier#009218220        UNITED KINGDOM
444218191   Manufacturer#1            qF  7NqTnlJAYwN4PUdJZxm95K
33-949-902-3942 warthogs haggle. ironic instructions according to the
furiously even warh
9999.920000             Supplier#012981983        GERMANY
207981970   Manufacturer#2            G
ZE9qAcKJsGNhHjqFjPBrvBni568YNLiQoP65i  17-433-196-6838 even pinto beans
of the f
9999.920000             Supplier#017930359        GERMANY
92930352    Manufacturer#3            oBAZ29DX327hdBqyT6BCFRBp8ytpFXA1o
17-976-148-2679 fluffily daring packages unwind furiously final
theodolites. silent, b
9999.920000             Supplier#013290934        RUSSIA
328290913
Manufacturer#2            ,Um0rw3k,VWVwG8uFfirFtmfkQB4kbVTH54scif  32-
557-549-3071 blithely silent instructions use slyly bold excuses.
unusual dolphins b
9999.910000             Supplier#016374772        ROMANIA
196374771   Manufacturer#2             D6OuxZ2jZD8KNWt
29-764-925-9071 requests integrate about the slyly express deposits.
carefully unusual id
9999.900000             Supplier#006303728        FRANCE
381303703   Manufacturer#4            gV9bRBGvnGQtL0C2mDS
16-529-728-8969 blithely special accounts integrate quickly. q
9999.880000             Supplier#006074547        ROMANIA
163574531   Manufacturer#5            IHU6ST78qQqIbbwg2ZKofC,btr9
29-648-701-3132 bold, silent platelets are along the slyly express
requests. close accounts within the fina
```

9999.880000 Supplier#021969178 RUSSIA
374469165 Manufacturer#3 qMaZcSRLZpekjgM,6w0LiB
32-822-390-9599 slyly final pearls cajole furiously. furiously regular
instructions accor
9999.870000 Supplier#019622191 FRANCE
372122178 Manufacturer#2 vUbSSdG lYaqbiHFf1NpUcO
uS4W1idES0Sw 16-963-449-7371 furiously final patterns despite the
furiously exp
9999.850000 Supplier#005466582 GERMANY
597966562 Manufacturer#3 aeBw1Bwblnr4p1WX0muFqwfC67m
17-634-657-2765 slyly even ideas use quickly blithely careful accounts.
regular, unusual packages mi
9999.830000 Supplier#020937869 ROMANIA
148437856 Manufacturer#1 6kpFr7EFtqhwgE Xk6IesOyoAa4KrgxXF
29-567-392-2176 quickly pending deposits could sleep quickly. special,
bold accou
9999.820000 Supplier#029345622 FRANCE
104345615 Manufacturer#2
DsBUvyaBvfV,jvNlULXXfLscUcAZiP0nlLz47a 16-144-835-2476 final platelets
boost above the regular packages. packages print furiously. regular
packag
9999.820000 Supplier#029345622 FRANCE
336845588 Manufacturer#3
DsBUvyaBvfV,jvNlULXXfLscUcAZiP0nlLz47a 16-144-835-2476 final platelets
boost above the regular packages. packages print furiously. regular
packag
9999.810000 Supplier#017667694 RUSSIA
10167693 Manufacturer#1 2A0Ybt5Wy2j1hax1yDWeEcUgeYX sRi
32-284-719-2965 furiously bold dependencies haggle after the furiously
special ideas. furiously even account
9999.810000 Supplier#017667694 RUSSIA
400167680 Manufacturer#3 2A0Ybt5Wy2j1hax1yDWeEcUgeYX sRi
32-284-719-2965 furiously bold dependencies haggle after the furiously
special ideas. furiously even account
9999.780000 Supplier#017756831 GERMANY
212756816 Manufacturer#2 4ah2OR5ECo8nXniZkp
17-825-175-3512 bold deposits cajole quickly beyond the final excuses.
pending, pendi
9999.780000 Supplier#017756831 GERMANY
482756798 Manufacturer#4 4ah2OR5ECo8nXniZkp
17-825-175-3512 bold deposits cajole quickly beyond the final excuses.
pending, pendi
9999.770000 Supplier#002254341 GERMANY
107254334 Manufacturer#5
s1galT4BQR2T7yPpV82IlktNcKI3Mm3Yt6I 17-391-701-5367 slyly ironic
deposits according to the evenly regular requests wake furiously over
the ir
9999.760000 Supplier#019287480 ROMANIA
281787470 Manufacturer#2 ASMs1elTGHVCbmSBus6yQfAwag
29-609-623-6842 slyly regular packages after the fluffily ironic
accounts na
9999.750000 Supplier#020636027 ROMANIA
373136014 Manufacturer#3 ckQQ63B64zuyLvax6F3qhR
29-632-416-4298 blithely even packages maintain blithely bold r
9999.700000 Supplier#000836396 ROMANIA
30836395 Manufacturer#2 Nkr gSMF3e4
29-579-143-3270 fluffily final dinos wake! final pinto beans sleep
furiously across the carefully idle requests. ir
9999.700000 Supplier#015136408 UNITED KINGDOM
247636399 Manufacturer#1 TZ7Lpnid0ILGdxhBCogEdV2Md 1,NaOCM4
33-760-859-6250 ironic platelets print boldly alongside of the pinto
beans. express theodoli
9999.690000 Supplier#013607282 FRANCE
238607267 Manufacturer#4 Ty7mHPSmn HXnSM
16-429-654-6108 quickly even multipliers according to the pending,
9999.690000 Supplier#021637376 RUSSIA
194137369 Manufacturer#1 vA3Xh6eAhNdFl5Cw5WVFh
32-236-447-4285 requests breach. carefully express accounts are
alongside of the final requests. special dolphins a
9999.670000 Supplier#019903981 UNITED KINGDOM
94903974 Manufacturer#4 ,JfHQX9UnRPnwyXBhdz9t
33-854-119-9479 even packages boost blithely. regular, pending pinto
beans sleep along
9999.660000 Supplier#009764817 GERMANY
377264780 Manufacturer#4 hpg8rJOp9l8troojbVhSplF406ll
17-156-291-3963 carefully express ideas sleep across the furiously final
foxes. express sauternes t
9999.650000 Supplier#000143654 FRANCE
232643646 Manufacturer#2
HhHCZ,RosE8He4uYvyIDqsPZe,7cZiJh1y9, 16-166-504-5864 pinto beans
wake blithely carefully final packages. bold reques
9999.650000 Supplier#027635360 FRANCE
297635359 Manufacturer#2 To4ymL1tJc4WP8Wba02bKuGS
16-327-348-7936 furiously bold dolphins within the blithely regular
requests are according to the careful accou
9999.650000 Supplier#011461806 GERMANY
78961799 Manufacturer#1 hD5FO3KgHMbsLuuX7Q2vYpRcw
17-909-480-6686 stealthy, unusual pearls sleep quickly busy, express ide
9999.620000 Supplier#013336935 RUSSIA
260836910 Manufacturer#1 ,b7GWe1qHR9IO 7Kyh6
32-100-814-3607 bravely ironic packages shall have to haggle ideas.
regular foxes are furious
9999.590000 Supplier#000781113 GERMANY
428281070 Manufacturer#4 ze212baTKt95P48t8J8Dr3Gaa
17-935-464-7550 regular deposits use pending foxes. ruthlessly even
platelets acro
9999.590000 Supplier#017327910 RUSSIA
354827876 Manufacturer#3 AhhoLwGuwQ,aM
32-585-326-6984 carefully ironic ideas wake slyly according to the
sometimes final accounts.
9999.580000 Supplier#005998677 ROMANIA
95998676 Manufacturer#4
9kDDRZLZNC8TH0YLR693qVhCc8lYgvrxOpufMaEa 29-270-812-7300 silent accounts
around the regular deposits sleep carefully about th
9999.570000 Supplier#007745978 GERMANY
247745977 Manufacturer#3 fJM4gx,wtmU
17-403-578-8945 regular, regular deposits use fluffily al
9999.560000 Supplier#010181476 FRANCE
160181475 Manufacturer#1 j70FFvvU3XPYVJzIDYZRDG9HgkPtyV,
16-473-375-7148 accounts are doggedly against the special requests.
regularly regular instructions

```
9999.560000              Supplier#001297944        GERMANY
353797932   Manufacturer#3
hD7olWLtWLKOn5gmphKNV1k1sfpYNsDBFoa5x1s  17-156-666-8873 slyly express
waters haggle bold ideas. blithely express dependencies are. reg
9999.560000              Supplier#020518148        GERMANY
388018111   Manufacturer#2            jRV0XqK8ulBYn9tFZJO8om
17-731-717-5164 carefully pending waters cajole slyly. carefully ironic
foxes use idly above the ironic di
9999.550000              Supplier#016359385        FRANCE
226359384   Manufacturer#3            AA2 qs7nzJiBxb8ab8
abZAulbwB5lnd72NoR4O8 16-126-600-9527 fluffily final requests sleep
slyly. even packages nag slyly. carefully regular deposits about
9999.550000              Supplier#020857343        FRANCE
343357331   Manufacturer#5
nGEzBxIRlsYGEiKVYk1bnq9E9drABkBm7JwHhW   16-746-317-4779 unusual, final
foxes boost. carefully
9999.540000              Supplier#004462825        FRANCE
244462824   Manufacturer#4            kk5J8urlRY
16-497-420-2053 quickly express accounts might ru
9999.520000              Supplier#020369811        GERMANY
365369786   Manufacturer#1            PMlyDgr95izDhJ5M7
17-337-829-3994 slyly pending accounts among the
9999.520000              Supplier#003573244        RUSSIA
513573243   Manufacturer#3
In,uLBaEjqQOrvstq603yTakfR42RU,L3rs7 M   32-954-902-3620 silent accounts
above the theodolites sleep furiously express asymptotes. ironic,
pending fox
9999.520000              Supplier#008256102        UNITED KINGDOM
420756087   Manufacturer#4            c5PCX58XtWLfIoOoH2
33-885-798-4231 final requests behind the ideas boost slyly i
9999.510000              Supplier#005796568        GERMANY
28296567    Manufacturer#1            UFBi5,y5EoFY
KdcRSj98pQJ1Q0l7NxZ9vy      17-616-190-2594 slyly final platelets affix
quickly-- requests cajole slyly at t
9999.510000              Supplier#005796568        GERMANY
470796537   Manufacturer#5            UFBi5,y5EoFY
KdcRSj98pQJ1Q0l7NxZ9vy      17-616-190-2594 slyly final platelets affix
quickly-- requests cajole slyly at t
9999.500000              Supplier#024090547        FRANCE
256590538   Manufacturer#5            kPpk6SUR7oq5hzSeqwreyE1spCg0
16-444-390-8471 unusual, even deposits haggle quickly. even, pending
deposits about the requests affix
9999.500000              Supplier#000730330        RUSSIA
555730293   Manufacturer#2
cWymp2YzIKwOIXA4zJwG72SvviR7IC8,ZCVS2    32-696-914-8119 fluffily final
multipliers use fluffy, pending foxes. quick,
9999.500000              Supplier#018003375        RUSSIA
370503362   Manufacturer#5            vb66v5OJ0VMpC1mLX
32-296-780-9452 accounts doze doggedly furiously bold deposits.
furiously silent pinto b
9999.490000              Supplier#010574001        GERMANY
445573972   Manufacturer#4
5xQ7,Yy,Ry62MEnAvpbpVtNj0HC9FccCh4hJ     17-594-416-6748 blithely
regular platelets thrash slyly. blithely
9999.460000              Supplier#019387217        ROMANIA
551887198   Manufacturer#5
y,TiQtwMPOI61QCyu4Qj7oHFl7XGGZAfRv24X    29-248-490-2372 daringly
regular instructions cajole
9999.460000              Supplier#029777398        ROMANIA
344777375   Manufacturer#4            fsDpZ wcbYR4U172cAO1DxTF
29-362-906-4168 furiously special ideas haggle furio
9999.450000              Supplier#003821847        GERMANY
243821846   Manufacturer#2            v 6i0T2jWhjuk6G
17-354-495-2396 theodolites affix blithely furio
9999.450000              Supplier#015558797        GERMANY
45558796    Manufacturer#4            g620ZFfSBVfg
17-530-776-2512 bold instructions haggle furiously across the even, bold
foxes. carefully bold theodolites impre
9999.450000              Supplier#025872183        ROMANIA
498372166   Manufacturer#3            eiFFSs9m6 Qu7yZ,EJqMv1jiyDOC1OM
29-550-167-2743 carefully final requests sleep blithely final deposits.
final theodolites haggle
9999.450000              Supplier#016420777        UNITED KINGDOM
181420764   Manufacturer#4            eLa7wPCUoVTU34xFxit
33-948-141-4140 furiously even foxes across the blithely ironic deposits
ought to serve blithely about the carefully
9999.430000              Supplier#026691790        UNITED KINGDOM
41691787    Manufacturer#1            76fD5xwbk9W96qir7m63x
33-604-823-4051 silent attainments wake ruthlessly unusual, regular
waters. slyly unusual accounts may use amo
9999.420000              Supplier#005702256        GERMANY
305702255   Manufacturer#4            dpHAwv dIa rB6C kdmRRyg42zgFkHFGs
17-671-279-6258 blithely express deposits haggle slyly. furiously regu
9999.420000              Supplier#006461057        GERMANY
531461022   Manufacturer#3            PhK4ObkJrx0m
17-195-919-3446 blithely final requests are quickly silent dependencies.
busy, silent platelets would lose. sil
9999.420000              Supplier#024807316        ROMANIA
152307300   Manufacturer#4            nXsvyv,IEMa
29-322-637-3752 blithely regular dinos wake above the quick
9999.410000              Supplier#012991690        GERMANY
387991665   Manufacturer#2            B3fhHo4R8yoYQJ0y
17-176-168-5559 blithely express theodolites hinder. quick excuses above
the quickly final frets are silently expres
9999.410000              Supplier#008701691        UNITED KINGDOM
121201686   Manufacturer#1            oYhc0rz80p9Pa6N
33-243-124-4905 furiously pending requests are slyly ironic requests.
furiously bold asympt
9999.400000              Supplier#012379945        GERMANY
214879937   Manufacturer#3            bp4A tvI 2
17-860-703-8908 slyly final dolphins haggle packages. dependencies above
the furio
9999.400000              Supplier#012291008        UNITED KINGDOM
72291007    Manufacturer#3            5x2d2hEVfJYaFMlg
33-859-778-9147 carefully pending requests according
9999.400000              Supplier#015789515        UNITED KINGDOM
98289511    Manufacturer#5
8BqceIogsui94jeIHtALwV3qZXyQUI9iCsKfk    33-651-613-8558 furiously
special platelets sleep blithely among the fluffily final theodolites.
slyly regular
9999.390000              Supplier#003375449        GERMANY
85875446    Manufacturer#5            Cn5rXsWVuHJac5HbdiZ6Je,ue9EJJGVg
```

```
17-488-708-4249 carefully even pinto beans sleep slyly. pending deposits
breach between the b
9999.370000              Supplier#014803396        FRANCE
277303386   Manufacturer#2            GNnja3cVcAzIJ4xU2V6x44aQs,FI
16-963-379-4498 even requests are slyly. fluffily final sheaves grow
furiously. express, ironic ideas
9999.370000              Supplier#004508001        ROMANIA
94508000    Manufacturer#3            b7N1xSmRer6VG
ouEx6aImfJStNi2gZSWjaEcAg  29-975-562-5264 fluffily regular deposits
along the slyly ironic packages are blithe
9999.370000              Supplier#004508001        ROMANIA
394508000   Manufacturer#3            b7N1xSmRer6VG
ouEx6aImfJStNi2gZSWjaEcAg  29-975-562-5264 fluffily regular deposits
along the slyly ironic packages are blithe
9999.370000              Supplier#020899603        ROMANIA
508399554   Manufacturer#4            PLQD74PzZQAJVJ,INr0VQ
29-818-693-4154 furiously final courts nag? silent ideas wake according
to the requests. regular, bold deposits
9999.350000              Supplier#017133555        RUSSIA
564633500   Manufacturer#5            ZF,q,jWF6a1cTNt9ho1so5VbJNUm2UUCt
32-205-304-4229 ironic, ironic theodolites wake c
9999.320000              Supplier#011190931        GERMANY
243690922   Manufacturer#1            H97aPeBZrbIV jw6WL3QeeJ K
17-936-626-8304 quickly special instructions wake furiously according to
the
9999.320000              Supplier#007812718        RUSSIA
187812717   Manufacturer#4             PFwuflgdlnZRWbVrJxkS,uE
79hbazrG2l      32-745-529-6725 slyly final instructions against the
special accounts engage regular requests: pin
9999.320000              Supplier#026837743        UNITED KINGDOM
371837718   Manufacturer#5            DcZXZfqAMYihanDbqY9q4MytI
33-516-503-9587 requests unwind about the special, even c
9999.320000              Supplier#029894766        UNITED KINGDOM
67394759    Manufacturer#3            cSxVTvuxiiPY8lZZruAx5Q,rTz bn0utBv
33-686-902-7041 regular deposits cajole furiously. furiously pending
instructions sleep carefully ironic excuses. fl
9999.320000              Supplier#029894766        UNITED KINGDOM
209894765   Manufacturer#5            cSxVTvuxiiPY8lZZruAx5Q,rTz bn0utBv
33-686-902-7041 regular deposits cajole furiously. furiously pending
instructions sleep carefully ironic excuses. fl
9999.310000              Supplier#011858739        GERMANY
581858738   Manufacturer#5
VwGBIkRbI8t9w2i5Ye8fN78fWIhOt180H4UGB    17-434-431-3407 slyly bold
accounts detect furiously. slyly pending requests use fluffily blithe
deposits.
9999.310000              Supplier#001620917        ROMANIA
279120889   Manufacturer#5            hIdzhMDB
Re5pu6xECcnq560OTgOYtksjxgvzH   29-262-936-4618 slyly even requests
haggle slyly. fluffily final ideas sleep. furiously even packages
according to
9999.310000              Supplier#001620917        ROMANIA
511620916   Manufacturer#2            hIdzhMDB
Re5pu6xECcnq560OTgOYtksjxgvzH   29-262-936-4618 slyly even requests
haggle slyly. fluffily final ideas sleep. furiously even packages
according to
9999.310000              Supplier#020235741        ROMANIA
492735724   Manufacturer#2
ImotdsoJBugpgub,clzR1x4lTAVSkAygQCc1 4   29-219-750-3851 furiously bold
deposits integrate carefully after the carefully regular accounts. quick
accounts
9999.310000              Supplier#020235741        ROMANIA
522735723   Manufacturer#5
ImotdsoJBugpgub,clzR1x4lTAVSkAygQCc1 4   29-219-750-3851 furiously bold
deposits integrate carefully after the carefully regular accounts. quick
accounts
9999.300000              Supplier#012758300        GERMANY
387758275   Manufacturer#5            nz 3WAv,OBARVBvODfEAIKv9N1
17-803-164-3122 carefully express accounts according to the furiousl
9999.300000              Supplier#020030608        GERMANY
260030607   Manufacturer#4            wVoOCSSEjDWrbp1dvdaW05YrR5
17-791-584-2548 final, daring dependencies inte
9999.290000              Supplier#013751117        FRANCE
21251116    Manufacturer#4
SQlf3S1MLMkQurYCe0ZWwrXInStooyLp81MxIx5w 16-239-731-2938 special, ironic
requests detect furiously furiously special
9999.290000              Supplier#023603167        FRANCE
23603166    Manufacturer#1            2uCZzsyXJgO2Pro
16-966-808-1669 slyly permanent pinto beans impress permanently. slyly
express accounts sleep boldly about the qui
9999.270000              Supplier#025186825        RUSSIA
550186788   Manufacturer#1            uTnDTSWGnN cPiF7ic9OBeATNYZoT
32-408-653-9247 slyly regular requests believe quickly furiously special
packages. carefully final p
9999.260000              Supplier#011231875        UNITED KINGDOM
56231872    Manufacturer#5            HwSH6Y9AAL io0tRwKFeIvozVYjG
33-269-740-4456 excuses integrate. even gifts sleep fluffily above the
slyly expr
9999.250000              Supplier#018724536        GERMANY
408724535   Manufacturer#4            GIcKFi8GOgaea3Nb49MroqA36LtKlnRkH
17-399-881-4969 slyly regular ideas use blithely along t
9999.240000              Supplier#017707404        UNITED KINGDOM
437707403   Manufacturer#3            26ewn9
gleBh1MLCKL5pEnz3TpXiYHTuJUkqDF   33-684-234-3555 special, express ideas
de
9999.230000              Supplier#004820971        GERMANY
132320958   Manufacturer#3            pHY,AVzjy0ReRZLO M
17-266-241-8135 regular, regular deposits sleep above the final requests.
quickly ironic theodolites cajol

(100 row(s) affected)
```

EXECUTE STREAM 0 QUERY 3

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 3 - Shipping Priority */

SELECT  TOP 10
```

```
        L_ORDERKEY,
        SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT))   AS REVENUE,
        O_ORDERDATE,
        O_SHIPPRIORITY
FROM    CUSTOMER,
        ORDERS,
        LINEITEM
WHERE   C_MKTSEGMENT    = 'HOUSEHOLD' AND
        C_CUSTKEY       = O_CUSTKEY AND
        L_ORDERKEY      = O_ORDERKEY AND
        O_ORDERDATE     < '1995-03-20' AND
        L_SHIPDATE      > '1995-03-20'
GROUP   BY      L_ORDERKEY,
                O_ORDERDATE,
                O_SHIPPRIORITY
ORDER   BY      REVENUE DESC,
                O_ORDERDATE
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 3 (MAX 200 ROWS)
------------------------------------------------------------

L_ORDERKEY           REVENUE                  O_ORDERDATE
O_SHIPPRIORITY
-------------------- ------------------------ ----------------------- --
------------
8582502496           524012.667700            1995-03-19 00:00:00.000 0
17172437062          524012.667700            1995-03-19 00:00:00.000 0
6783327621           513221.829100            1995-03-01 00:00:00.000 0
15373262211          513221.829100            1995-03-01 00:00:00.000 0
4054421287           510678.441300            1995-03-14 00:00:00.000 0
12644355877          510678.441300            1995-03-14 00:00:00.000 0
131698920            507524.133800            1995-03-09 00:00:00.000 0
8145113669           505741.627000            1995-03-04 00:00:00.000 0
16735048259          505741.627000            1995-03-04 00:00:00.000 0
13282498022          505629.915400            1995-02-25 00:00:00.000 0

(10 row(s) affected)
```

EXECUTE STREAM 0 QUERY 4

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 4 - Order Priority Checking */

SELECT  O_ORDERPRIORITY,
        COUNT(*)              AS ORDER_COUNT
FROM    ORDERS
WHERE   O_ORDERDATE    >= '1996-02-01' AND
        O_ORDERDATE    < dateadd (mm, 3, '1996-02-01') AND
        EXISTS         (      SELECT  *
                              FROM    LINEITEM
                              WHERE   L_ORDERKEY     = O_ORDERKEY AND
                                      L_COMMITDATE   < L_RECEIPTDATE
                      )
GROUP   BY      O_ORDERPRIORITY
ORDER   BY      O_ORDERPRIORITY
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 4 (MAX 200 ROWS)
------------------------------------------------------------

O_ORDERPRIORITY ORDER_COUNT
--------------- -----------
1-URGENT        30899496
2-HIGH          30904973
3-MEDIUM        30901072
4-NOT SPECIFIED 30900316
5-LOW           30902280

(5 row(s) affected)
```

EXECUTE STREAM 0 QUERY 5

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 5 - Local Supplier Volume */

SELECT  N_NAME,
        SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT))   AS REVENUE
FROM    CUSTOMER,
        ORDERS,
        LINEITEM,
        SUPPLIER,
        NATION,
        REGION
WHERE   C_CUSTKEY       = O_CUSTKEY AND
        L_ORDERKEY      = O_ORDERKEY AND
        L_SUPPKEY       = S_SUPPKEY AND
        C_NATIONKEY     = S_NATIONKEY AND
        S_NATIONKEY     = N_NATIONKEY AND
        N_REGIONKEY     = R_REGIONKEY AND
        R_NAME          = 'MIDDLE EAST' AND
        O_ORDERDATE     >= '1997-01-01' AND
        O_ORDERDATE     < DATEADD(YY, 1, '1997-01-01')
GROUP   BY      N_NAME
ORDER   BY      REVENUE DESC
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 5 (MAX 200 ROWS)
```

```
------------------------------------------------------------

N_NAME                    REVENUE
------------------------- -------------------------
IRAN                      159138762929.012880
EGYPT                     159026098338.004300
IRAQ                      158993393647.031160
JORDAN                    158973094946.062930
SAUDI ARABIA              158950883459.440090

(5 row(s) affected)
```

EXECUTE STREAM 0 QUERY 6

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 6 - Forecasting Revenue Change */

SELECT  SUM(L_EXTENDEDPRICE*L_DISCOUNT)      AS REVENUE
FROM    LINEITEM
WHERE   L_SHIPDATE     >= '1997-01-01' AND
        L_SHIPDATE     < dateadd (yy, 1, '1997-01-01') AND
        L_DISCOUNT     BETWEEN 0.04 - 0.01 AND 0.04 + 0.01 AND
        L_QUANTITY     < 24
```

OUTPUT OF EXECUTE STREAM 0 QUERY 6 (MAX 200 ROWS)

```
------------------------------------------------------------

REVENUE
------------------------
246831750235.501920

(1 row(s) affected)
```

EXECUTE STREAM 0 QUERY 7

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 7 - Volume Shipping */

SELECT  SUPP_NATION,
        CUST_NATION,
        L_YEAR,
        SUM(VOLUME)     AS REVENUE
FROM    (       SELECT  N1.N_NAME                   AS SUPP_NATION,
                        N2.N_NAME                   AS CUST_NATION,
                        datepart(yy,L_SHIPDATE)             AS L_YEAR,
                        L_EXTENDEDPRICE*(1-L_DISCOUNT)      AS VOLUME
                FROM    SUPPLIER,
                        LINEITEM,
                        ORDERS,
                        CUSTOMER,
                        NATION N1,
                        NATION N2
                WHERE   S_SUPPKEY       = L_SUPPKEY AND
                        O_ORDERKEY      = L_ORDERKEY AND
                        C_CUSTKEY       = O_CUSTKEY AND
                        S_NATIONKEY     = N1.N_NATIONKEY AND
                        C_NATIONKEY     = N2.N_NATIONKEY AND
                        (       (N1.N_NAME      = 'FRANCE'      AND
N2.N_NAME       = 'CANADA')
                                OR
                                (N1.N_NAME      = 'CANADA'      AND
N2.N_NAME       = 'FRANCE')
                        ) AND
                        L_SHIPDATE      BETWEEN '1995-01-01' AND '1996-12-
31'
        )       AS SHIPPING
GROUP   BY      SUPP_NATION,
                CUST_NATION,
                L_YEAR
ORDER   BY      SUPP_NATION,
                CUST_NATION,
                L_YEAR
```

OUTPUT OF EXECUTE STREAM 0 QUERY 7 (MAX 200 ROWS)

```
------------------------------------------------------------

SUPP_NATION               CUST_NATION               L_YEAR      REVENUE
------------------------- ------------------------- ----------- --------
----------------
CANADA                    FRANCE                    1995
159009989774.475010
CANADA                    FRANCE                    1996
159559354480.610570
FRANCE                    CANADA                    1995
158813396788.655300
FRANCE                    CANADA                    1996
159286435011.937990

(4 row(s) affected)
```

EXECUTE STREAM 0 QUERY 8

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 8 - National Market Share */
```

```
SELECT  O_YEAR,
        SUM(CASE        WHEN    NATION  = 'CANADA'
                        THEN    VOLUME
                        ELSE    0
                        END) / SUM(VOLUME)     AS MKT_SHARE
FROM    (       SELECT  datepart(yy,O_ORDERDATE)              AS O_YEAR,
                        L_EXTENDEDPRICE * (1-L_DISCOUNT)      AS VOLUME,
                        N2.N_NAME                             AS NATION
                FROM    PART,
                        SUPPLIER,
                        LINEITEM,
                        ORDERS,
                        CUSTOMER,
                        NATION N1,
                        NATION N2,
                        REGION
                WHERE   P_PARTKEY      = L_PARTKEY AND
                        S_SUPPKEY      = L_SUPPKEY AND
                        L_ORDERKEY     = O_ORDERKEY AND
                        O_CUSTKEY      = C_CUSTKEY AND
                        C_NATIONKEY    = N1.N_NATIONKEY AND
                        N1.N_REGIONKEY = R_REGIONKEY AND
                        R_NAME         = 'AMERICA' AND
                        S_NATIONKEY    = N2.N_NATIONKEY AND
                        O_ORDERDATE    BETWEEN '1995-01-01' AND '1996-12-
31' AND
                        P_TYPE         = 'SMALL ANODIZED STEEL'
        )       AS      ALL_NATIONS
GROUP   BY      O_YEAR
ORDER   BY      O_YEAR
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 8 (MAX 200 ROWS)
------------------------------------------------------------

O_YEAR      MKT_SHARE
----------- ------------------------
1995        0.040129
1996        0.039886

(2 row(s) affected)
```

EXECUTE STREAM 0 QUERY 9

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 9 - Product Type Profit Measure */

SELECT  NATION,
        O_YEAR,
        SUM(AMOUNT)    AS SUM_PROFIT
FROM    (       SELECT  N_NAME
        AS NATION,
                        datepart(yy, O_ORDERDATE)
        AS O_YEAR,
                        L_EXTENDEDPRICE*(1-L_DISCOUNT)-
PS_SUPPLYCOST*L_QUANTITY      AS AMOUNT
                FROM    PART,
                        SUPPLIER,
                        LINEITEM,
                        PARTSUPP,
                        ORDERS,
                        NATION
                WHERE   S_SUPPKEY      = L_SUPPKEY AND
                        PS_SUPPKEY     = L_SUPPKEY AND
                        PS_PARTKEY     = L_PARTKEY AND
                        P_PARTKEY      = L_PARTKEY AND
                        O_ORDERKEY     = L_ORDERKEY AND
                        S_NATIONKEY    = N_NATIONKEY AND
                        P_NAME         LIKE '%cornflower%'
        )       AS PROFIT
GROUP   BY      NATION,
                O_YEAR
ORDER   BY      NATION,
                O_YEAR  DESC
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 9 (MAX 200 ROWS)
------------------------------------------------------------

NATION                    O_YEAR      SUM_PROFIT
------------------------- ----------- -------------------------
ALGERIA                   1998        99359510174.901138
ALGERIA                   1997        169290934413.747470
ALGERIA                   1996        169817979259.643040
ALGERIA                   1995        169180193696.551670
ALGERIA                   1994        169143657539.004670
ALGERIA                   1993        169181078660.423070
ALGERIA                   1992        169896678598.759580
ARGENTINA                 1998        99234550986.603973
ARGENTINA                 1997        169102076904.238370
ARGENTINA                 1996        169577950990.629330
ARGENTINA                 1995        169148814728.310210
ARGENTINA                 1994        169291918634.413210
ARGENTINA                 1993        169269894148.738980
ARGENTINA                 1992        169875176461.224060
BRAZIL                    1998        99166140832.922470
BRAZIL                    1997        169043532930.510710
BRAZIL                    1996        169542206315.641330
BRAZIL                    1995        169090254553.919250
BRAZIL                    1994        169256947129.304530
BRAZIL                    1993        169160633194.899230
BRAZIL                    1992        169445763946.608030
CANADA                    1998        99524675700.557831
```

CANADA	1997	169554982893.278170	IRAN	1997	169185489133.717100
CANADA	1996	169782887457.506500	IRAN	1996	169646385995.076260
CANADA	1995	169491964756.959530	IRAN	1995	169099905035.643710
CANADA	1994	169515054923.346440	IRAN	1994	169376532774.695650
CANADA	1993	169524008638.681640	IRAN	1993	169164289790.246830
CANADA	1992	170103840700.290800	IRAN	1992	169723640877.812130
CHINA	1998	99361122656.110138	IRAQ	1998	99261845362.722168
CHINA	1997	169280102377.311490	IRAQ	1997	169730378090.197270
CHINA	1996	169652214260.766940	IRAQ	1996	169818155117.856480
CHINA	1995	169256608720.106380	IRAQ	1995	169592064761.059780
CHINA	1994	169314759115.569090	IRAQ	1994	169139004689.326080
CHINA	1993	169324131449.656250	IRAQ	1993	169263032488.068850
CHINA	1992	169606395165.547670	IRAQ	1992	169986378535.705570
EGYPT	1998	99748562506.608475	JAPAN	1998	99042365054.581879
EGYPT	1997	169783775222.282500	JAPAN	1997	169192746471.660950
EGYPT	1996	170468882044.618870	JAPAN	1996	169623443351.726590
EGYPT	1995	169582328683.521580	JAPAN	1995	169200558972.874420
EGYPT	1994	169777406073.194430	JAPAN	1994	169083608833.893680
EGYPT	1993	169915020705.166900	JAPAN	1993	169149192509.353490
EGYPT	1992	170245060814.222140	JAPAN	1992	169750705945.043520
ETHIOPIA	1998	99060760752.314468	JORDAN	1998	99036424186.232285
ETHIOPIA	1997	169010389275.117610	JORDAN	1997	168838270428.625240
ETHIOPIA	1996	169604208256.395260	JORDAN	1996	169567928716.189120
ETHIOPIA	1995	169062358640.225740	JORDAN	1995	168812866982.614690
ETHIOPIA	1994	169242820658.765230	JORDAN	1994	169119742834.061190
ETHIOPIA	1993	169251993261.854190	JORDAN	1993	168825893945.043120
ETHIOPIA	1992	169687791644.761170	JORDAN	1992	169409810433.523190
FRANCE	1998	99267623644.048767	KENYA	1998	99354828430.159958
FRANCE	1997	169295807188.673160	KENYA	1997	169273177958.326230
FRANCE	1996	169768600136.830290	KENYA	1996	169543801980.572510
FRANCE	1995	169270457265.581510	KENYA	1995	169213181344.990330
FRANCE	1994	169410766881.311520	KENYA	1994	169147712779.664280
FRANCE	1993	169269588508.119570	KENYA	1993	169328015212.427830
FRANCE	1992	169546116583.302340	KENYA	1992	169608075403.956670
GERMANY	1998	99176434404.959152	MOROCCO	1998	99266599187.743652
GERMANY	1997	169336206184.821560	MOROCCO	1997	169151477428.717350
GERMANY	1996	169944665902.148860	MOROCCO	1996	169611398419.792450
GERMANY	1995	169604465629.066410	MOROCCO	1995	169284206143.697480
GERMANY	1994	169436760348.177830	MOROCCO	1994	169079108295.432770
GERMANY	1993	169260276769.556120	MOROCCO	1993	168849464186.759770
GERMANY	1992	169625007914.217560	MOROCCO	1992	169617506827.327360
INDIA	1998	99052168011.650620	MOZAMBIQUE	1998	99043455808.567169
INDIA	1997	169066937166.496490	MOZAMBIQUE	1997	169156889748.409480
INDIA	1996	169365695081.258180	MOZAMBIQUE	1996	169576665776.907960
INDIA	1995	168976311570.226990	MOZAMBIQUE	1995	169353144656.096620
INDIA	1994	169018459296.382510	MOZAMBIQUE	1994	169202262340.588440
INDIA	1993	169062985877.954130	MOZAMBIQUE	1993	169349904804.448060
INDIA	1992	169394460413.796420	MOZAMBIQUE	1992	169559507656.834810
INDONESIA	1998	99252951689.100067	PERU	1998	99085700051.642517
INDONESIA	1997	169135800551.476960	PERU	1997	169126208251.490230
INDONESIA	1996	169686668335.200840	PERU	1996	169618838611.996920
INDONESIA	1995	169043178897.192110	PERU	1995	169118654237.688840
INDONESIA	1994	169241239918.471500	PERU	1994	168982833827.228270
INDONESIA	1993	169176227127.859160	PERU	1993	168909995792.333740
INDONESIA	1992	169586495071.937900	PERU	1992	169545126061.732970
IRAN	1998	99083566104.292572	ROMANIA	1998	99206414257.985886

```
ROMANIA                   1997        169051413913.962950
ROMANIA                   1996        169698455171.309050
ROMANIA                   1995        169089729379.615840
ROMANIA                   1994        169187482479.019170
ROMANIA                   1993        169164550423.286220
ROMANIA                   1992        169767436471.335880
RUSSIA                    1998        99093523383.168976
RUSSIA                    1997        168993752893.420900
RUSSIA                    1996        169199231955.840910
RUSSIA                    1995        168640307849.098050
RUSSIA                    1994        168766561900.246400
RUSSIA                    1993        168983029471.085570
RUSSIA                    1992        169436615560.461850
SAUDI ARABIA              1998        99197826658.211716
SAUDI ARABIA              1997        169161222283.117680
SAUDI ARABIA              1996        169662379451.071720
SAUDI ARABIA              1995        169031568552.398620
SAUDI ARABIA              1994        169055218371.439360
SAUDI ARABIA              1993        168945216772.707860
SAUDI ARABIA              1992        169695741320.880490
UNITED KINGDOM            1998        99090947951.274979
UNITED KINGDOM            1997        168861218956.585080
UNITED KINGDOM            1996        169596748529.884980
UNITED KINGDOM            1995        169166009752.684360
UNITED KINGDOM            1994        169181087251.357360
UNITED KINGDOM            1993        168916664614.417480
UNITED KINGDOM            1992        169502810870.425750
UNITED STATES             1998        99351290981.792618
UNITED STATES             1997        169210029467.050450
UNITED STATES             1996        169849509779.304380
UNITED STATES             1995        169373930209.773470
UNITED STATES             1994        169321092003.899050
UNITED STATES             1993        169625301975.914550
UNITED STATES             1992        169590942664.221530
VIETNAM                   1998        99458140130.466339
VIETNAM                   1997        169421521745.498750
VIETNAM                   1996        170128619566.631470
VIETNAM                   1995        169550873422.875000
VIETNAM                   1994        169559313103.146030
VIETNAM                   1993        169681382118.618840
VIETNAM                   1992        170284332084.305240

(175 row(s) affected)
```

EXECUTE STREAM 0 QUERY 10

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 10 - Returned Item Reporting */

SELECT  TOP 20
        C_CUSTKEY,
        C_NAME,
        SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT))   AS REVENUE,
        C_ACCTBAL,
        N_NAME,
        C_ADDRESS,
        C_PHONE,
        C_COMMENT
FROM    CUSTOMER,
        ORDERS,
        LINEITEM,
        NATION
WHERE   C_CUSTKEY       = O_CUSTKEY             AND
        L_ORDERKEY      = O_ORDERKEY            AND
        O_ORDERDATE     >= '1993-06-01'                       AND
        O_ORDERDATE     < dateadd(mm, 3, '1993-06-01')        AND
        L_RETURNFLAG    = 'R'                   AND
        C_NATIONKEY     = N_NATIONKEY
GROUP   BY      C_CUSTKEY,
                C_NAME,
                C_ACCTBAL,
                C_PHONE,
                N_NAME,
                C_ADDRESS,
                C_COMMENT
ORDER   BY      REVENUE DESC



OUTPUT OF EXECUTE STREAM 0 QUERY 10 (MAX 200 ROWS)
------------------------------------------------------------

C_CUSTKEY   C_NAME                    REVENUE                   C_ACCTBAL
N_NAME                    C_ADDRESS
C_PHONE         C_COMMENT
----------- ------------------------- ------------------------- ---------
--------------- ------------------------- ------------------------------
---------- --------------- ----------------------------------------------
--------------------------------------------------------------------------
80401750    Customer#080401750        1867295.373600
9739.350000             IRAN
HYD6QaktKy5inWsOj9vEB4G5NkS0NKW3Y0SeAezr 20-383-723-6008 furiously final
requests cajole. slyly pending platelets integrate blithely slyly even
pinto bean
350354287   Customer#350354287        1746297.091500
5189.180000             INDIA                     5GFB1KFighLX
18-940-864-6881 accounts detect quickly slyly unusual accounts. regular,
regular accounts wake furiously. accounts wake furiou
5838199     Customer#005838199        1731036.374400
6544.300000             IRAQ                      RpoAPcpDG4D
HXteW2tJfNaG                21-926-362-1545 blithely regular packages
was blithely. carefully ironic realms haggle furiously
321779422   Customer#321779422        1722623.627700
6127.080000             GERMANY
Xs0aVA4KXpv2qIswi1hoHXtEt2YDSPp          17-602-396-6708 carefully
pending pinto beans haggle toward the unusual pinto beans. slyly final
```

```
229958704   Customer#229958704        1707601.670400
8156.470000              FRANCE
vjDwWHDkTQLy6OpTseWVJG0gnsEnzIWe0nzUEb   16-960-526-1339 express hockey
players are quickly regular, express instruction
433608500   Customer#433608500        1692104.233800
8900.480000              PERU
ePoY47cQ,kLBWD6heUXB,ugn1                27-756-488-1904 furiously
silent deposits cajole carefully about the fluffily regular packages.
44414609    Customer#044414609        1691007.468300
9001.430000              JORDAN
moKdBd1vly40P4TcaXOSe1lWDUgLc            23-803-509-7670 even forges nag
furiously around the final instructions. ex
310748725   Customer#310748725        1687852.035000
3200.750000              JAPAN                     HnV
1twus4cyofisvTO5HV5TW4IPp72Zxv       22-495-593-5672 pending, regular
realms unwind requests. qu
444121453   Customer#444121453        1680738.853600           -
51.870000              JORDAN
R1DGaerTs6u3NAaJdGW7IPrh86zGxqNKbO2      23-298-423-7489 silent, regular
excuses wake. final ideas along the furiously even theodolites affix
stealthily
129304219   Customer#129304219        1673643.352900
4689.940000              GERMANY                    OX60 H4axnvfnsV7
17-803-996-8894 pinto beans along the final accounts are carefully
blithely final request
433124570   Customer#433124570        1669324.193400
1639.150000              ROMANIA
tcWcG1E2dfdutLE,RzbQd4XQJrbMLmSa         29-880-790-4933 pinto beans
detect slyly along the deposits. blithely ironic packages
230881475   Customer#230881475        1667158.889600
1578.730000              ALGERIA
0BHDdGyImVDZGhyqY9PiRmpzZETWn1wmlKPqm7g  10-581-161-2708 final packages
use slyly. even accounts after the carefully unusual packages
356139968   Customer#356139968        1649650.591200
4473.040000              FRANCE                    O
gSXm3IxjbG7ZSX1ohqSyxY8CD              16-289-120-1813 regular, unusual
pinto beans breach furiously special, special packages. quickly even
accounts are caref
65523640    Customer#065523640        1648532.375400
3177.820000              JAPAN                     fVJg2vVXeTFfwuu
22-520-928-4400 regular instructions about the blithely fluffy accoun
22378201    Customer#022378201        1646866.299000
4826.570000              RUSSIA                    OBQLBkCudzQRyyDT
EK7qS                   32-639-324-6077 silent, express accounts haggle
blithely even e
448416889   Customer#448416889        1640961.663000
5943.140000              ETHIOPIA                  H2sVNGi3qghQomEBh9jR
15-948-632-2315 fluffily regular depths along the blithely daring
asymptotes wake fluffily above the depen
282651241   Customer#282651241        1639584.627400
288.130000               UNITED KINGDOM
3qvNfWBKlkpRk9qiek27OwJcMJiBa5           33-853-401-4613 carefully
ironic foxes ought to nag
390569422   Customer#390569422        1635654.224800
2959.030000              VIETNAM                   L
nd,ESOEfw9lPHR7KtykUO1Kogfz4GkjW1zAlfX 31-991-703-4970 furiously silent
deposits sleep darin
304743842   Customer#304743842        1632724.986600           55.040000
ARGENTINA                bLgabjHoQODooC8LZDPt9EQQFNb             11-
307-732-7563 slyly unusual instructions nag carefully alongside of the
express attainments. slyly regular packages ca
240772040   Customer#240772040        1628304.525600
6647.110000              PERU
mTqDni3,FSUqRjUJtdTbOHOnmHKS96Rh8ea      27-844-446-1663 carefully
express instructions along the fluffily ironic accounts haggle fu

(20 row(s) affected)
```

EXECUTE STREAM 0 QUERY 11

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 11 - Important Stock Indentification */

SELECT  PS_PARTKEY,
        SUM(PS_SUPPLYCOST*PS_AVAILQTY)        AS VALUE
FROM    PARTSUPP,
        SUPPLIER,
        NATION
WHERE   PS_SUPPKEY     = S_SUPPKEY    AND
        S_NATIONKEY    = N_NATIONKEY  AND
        N_NAME         = 'ARGENTINA'
GROUP   BY      PS_PARTKEY
HAVING  SUM(PS_SUPPLYCOST*PS_AVAILQTY) >
                (       SELECT  SUM(PS_SUPPLYCOST*PS_AVAILQTY) *
0.0000000333
                        FROM    PARTSUPP,
                                SUPPLIER,
                                NATION
                        WHERE   PS_SUPPKEY     = S_SUPPKEY    AND
                                S_NATIONKEY    = N_NATIONKEY  AND
                                N_NAME         = 'ARGENTINA'
                )
ORDER   BY      VALUE   DESC
```

OUTPUT OF EXECUTE STREAM 0 QUERY 11 (MAX 200 ROWS)

```
PS_PARTKEY  VALUE
----------- ------------------------
156133815   27381188.280000
545975385   26390605.520000
414639777   26211795.950000
135749841   26009676.680000
408216584   25922596.050000
```

```
336535608   25907263.240000
124422979   25521365.810000
20274137    25193090.010000
526404001   24929364.300000
17874191    24921395.390000
503342404   24891490.000000
386718681   24780212.320000
443949238   24737346.520000
556246572   24698697.280000
252433341   24497206.630000
130815869   24371599.570000
446427310   24289575.840000
385148572   24268000.240000
147370417   24206306.380000
335304561   24183005.170000
31342938    24179841.880000
352708462   24136748.940000
117044716   23939453.660000
369384997   23804797.100000
929550      23738420.490000
195517471   23725380.500000
238945240   23592504.150000
501356670   23583615.700000
169286061   23550104.740000
368290425   23543163.110000
214768236   23539031.330000
163232398   23518758.710000
358014115   23515050.960000
590623806   23495337.800000
565442912   23398114.310000
321555622   23367949.160000
196505743   23360188.330000
384577326   23344774.000000
142532415   23342356.880000
51648794    23313038.500000
141816241   23190039.150000
152322536   23186780.520000
37425589    22976860.300000
22890437    22916834.290000
583079138   22886165.360000
139469973   22869786.450000
518693381   22844700.610000
522990927   22779127.760000
525243420   22745026.110000
415494561   22685833.160000
217465127   22647906.310000
106412024   22637011.200000
347340847   22617857.810000
463815249   22598312.390000
475061187   22579907.860000
253395760   22493741.140000
169234483   22484243.830000
16612437    22477335.600000
272925486   22422621.360000
359452791   22351415.240000
249945578   22321800.660000
434841630   22306464.250000
191834872   22277544.790000
480160060   22267653.990000
226223013   22261585.320000
460111366   22200878.210000
439555027   22189843.270000
464751433   22142060.280000
544641168   22109424.100000
87525298    22103192.440000
351806102   22090699.920000
35459902    22071419.240000
67051513    22056509.430000
188039085   21983729.230000
483169090   21973445.300000
475474394   21962755.120000
208840489   21960183.470000
546301529   21944403.140000
343194469   21938432.050000
268955520   21912533.070000
82748281    21909415.060000
304025583   21897945.490000
230846678   21878850.010000
378754890   21875224.600000
100846523   21862015.220000
528154537   21847009.850000
395324024   21833887.240000
415602604   21830941.760000
148305464   21820761.410000
389045676   21804364.630000
405219941   21791582.460000
23723890    21787685.610000
391729624   21765943.120000
542430653   21757821.130000
535800145   21756777.530000
133932779   21738839.430000
334333045   21736903.440000
45918258    21717548.440000
115262386   21669361.560000
493128811   21655137.390000

[2816069 more rows]

392342847   8003688.840000
412299856   8003688.300000
165034957   8003688.000000
366911456   8003687.890000
304346533   8003687.720000
236043036   8003687.340000
371992545   8003687.160000
120525939   8003686.950000
259910962   8003685.690000
8444864     8003685.600000
338881513   8003685.240000
50130110    8003684.130000
330816067   8003683.740000
208922411   8003683.710000
```

```
1144870     8003681.980000
47234596    8003681.700000
348577110   8003681.600000
270948692   8003681.600000
228019035   8003680.960000
403844399   8003680.880000
418307969   8003680.520000
509939526   8003680.520000
547850885   8003680.390000
9562401     8003679.840000
587828592   8003679.600000
487298853   8003679.180000
458646770   8003678.100000
175640807   8003677.500000
180050837   8003676.900000
379787290   8003676.720000
141943464   8003675.680000
145992558   8003675.680000
107123037   8003675.640000
569602102   8003675.500000
297079312   8003674.160000
504433377   8003674.120000
505567672   8003673.800000
404269488   8003673.100000
454879556   8003672.300000
204127258   8003672.100000
490164062   8003672.100000
71440011    8003672.040000
401844359   8003671.980000
71053300    8003671.910000
228142511   8003671.740000
352397466   8003670.020000
547506495   8003669.640000
224478729   8003668.680000
575197891   8003668.630000
227424514   8003667.600000
214834208   8003667.420000
78996777    8003667.200000
413723685   8003667.000000
313732935   8003666.770000
573663436   8003666.530000
217418617   8003666.160000
564556293   8003666.080000
411132149   8003665.950000
514819682   8003665.920000
433084327   8003665.570000
554120780   8003665.120000
519346247   8003664.780000
424698770   8003664.570000
500199362   8003664.420000
449757580   8003663.880000
215043660   8003663.740000
123879047   8003663.310000
221608548   8003662.870000
584137687   8003661.600000
287035038   8003661.180000
427197112   8003660.120000
458031814   8003659.780000
102074354   8003659.770000
477923501   8003659.590000
383914372   8003659.290000
504412857   8003659.290000
156815194   8003659.220000
373396041   8003659.180000
154525602   8003657.790000
509661362   8003657.710000
554573257   8003657.220000
120938176   8003657.200000
2716496     8003656.990000
362617059   8003656.800000
258230937   8003656.800000
26815953    8003656.800000
42868484    8003656.800000
558092376   8003656.760000
214456374   8003656.530000
291366607   8003656.360000
373387336   8003655.360000
313828400   8003655.080000
198724062   8003655.000000
562762267   8003655.000000
52649826    8003655.000000
420004531   8003654.400000
327360533   8003654.100000
258959621   8003653.840000
393521412   8003653.570000
224854287   8003653.390000

(2816269 row(s) affected)
```

EXECUTE STREAM 0 QUERY 12

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 12 - Shipping Modes and Order Priority */

SELECT  L_SHIPMODE,
        SUM(    CASE    WHEN O_ORDERPRIORITY  = '1-URGENT'    OR
                             O_ORDERPRIORITY  = '2-HIGH'
                        THEN 1
                        ELSE 0
                END)    AS HIGH_LINE_COUNT,
        SUM(    CASE    WHEN O_ORDERPRIORITY <> '1-URGENT'    AND
                             O_ORDERPRIORITY <> '2-HIGH'
                        THEN 1
                        ELSE 0
                END)    AS LOW_LINE_COUNT
FROM    ORDERS,
        LINEITEM
WHERE   O_ORDERKEY      = L_ORDERKEY          AND
```

```
        L_SHIPMODE     IN ('RAIL','FOB')             AND
        L_COMMITDATE   < L_RECEIPTDATE               AND
        L_SHIPDATE     < L_COMMITDATE        AND
        L_RECEIPTDATE  >= '1994-01-01'                        AND
        L_RECEIPTDATE  < dateadd(yy, 1, '1994-01-01')
GROUP   BY      L_SHIPMODE
ORDER   BY      L_SHIPMODE
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 12 (MAX 200 ROWS)
------------------------------------------------------------

L_SHIPMODE HIGH_LINE_COUNT LOW_LINE_COUNT
---------- --------------- --------------
FOB        18716257        28073263
RAIL       18720513        28069902

(2 row(s) affected)
```

EXECUTE STREAM 0 QUERY 13

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 13 - Customer Distribution */

SELECT  C_COUNT,
        COUNT(*)       AS CUSTDIST
FROM    (       SELECT  C_CUSTKEY,
                        COUNT(O_ORDERKEY)
                FROM    CUSTOMER left outer join ORDERS on
                        C_CUSTKEY      = O_CUSTKEY           AND
                        O_COMMENT      not like '%pending%deposits%'
                GROUP   BY      C_CUSTKEY
        )       AS C_ORDERS (C_CUSTKEY, C_COUNT)
GROUP   BY      C_COUNT
ORDER   BY      CUSTDIST       DESC,
                C_COUNT        DESC
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 13 (MAX 200 ROWS)
------------------------------------------------------------

C_COUNT     CUSTDIST
----------- -----------
0           150000000
10          66891106
20          38975220
11          33541771
19          29909842
21          29679272
18          27733308
8           21933315
9           17425689
22          12844196
12          7027850
17          5937787
23          3492311
7           2285966
13          743910
16          654473
24          636726
6           107246
25          80622
15          47134
14          41715
26          7062
5           2952
27          458
4           50
28          18
29          1

(27 row(s) affected)
```

EXECUTE STREAM 0 QUERY 14

```
------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 14 - Promotion Effect */

SELECT  100.00 * SUM   (       CASE    WHEN P_TYPE LIKE 'PROMO%'
                                       THEN L_EXTENDEDPRICE*(1-L_DISCOUNT)
                                       ELSE 0
                               END) / SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT))
        AS PROMO_REVENUE
FROM    LINEITEM,
        PART
WHERE   L_PARTKEY      = P_PARTKEY    AND
        L_SHIPDATE     >= '1997-02-01'               AND
        L_SHIPDATE     < dateadd(mm, 1, '1997-02-01')
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 14 (MAX 200 ROWS)
------------------------------------------------------------

PROMO_REVENUE
-------------------------
16.673567

(1 row(s) affected)
```

EXECUTE STREAM 0 QUERY 15

```
-------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 15 - Create View for Top Supplier Query */

CREATE  VIEW REVENUE0 (SUPPLIER_NO, TOTAL_REVENUE)
AS
SELECT  L_SUPPKEY,
        SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT))
FROM    LINEITEM
WHERE   L_SHIPDATE     >= '1994-03-01' AND
        L_SHIPDATE     < dateadd(mm, 3, '1994-03-01')
GROUP   BY      L_SUPPKEY
GO


/* TPC_H  Query 15 - Top Supplier */

SELECT  S_SUPPKEY,
        S_NAME,
        S_ADDRESS,
        S_PHONE,
        TOTAL_REVENUE
FROM    SUPPLIER,
        REVENUE0
WHERE   S_SUPPKEY      = SUPPLIER_NO AND
        TOTAL_REVENUE  = (     SELECT  MAX(TOTAL_REVENUE)
                               FROM    REVENUE0
                         )
ORDER   BY      S_SUPPKEY

DROP VIEW REVENUE0



OUTPUT OF EXECUTE STREAM 0 QUERY 15 (MAX 200 ROWS)
-------------------------------------------------------------

S_SUPPKEY   S_NAME                    S_ADDRESS
S_PHONE         TOTAL_REVENUE
----------- ------------------------- ---------------------------------
------ --------------- -------------------------
15272952    Supplier#015272952        4vKdLLK3MDBRK
12-194-545-2031 3672898.388400

(1 row(s) affected)
```

EXECUTE STREAM 0 QUERY 16

```
-------------------------------------------------------------

-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 16 - Parts/Supplier Relationship */

SELECT  P_BRAND,
        P_TYPE,
        P_SIZE,
        COUNT(DISTINCT PS_SUPPKEY)    AS SUPPLIER_CNT
FROM    PARTSUPP,
        PART
WHERE   P_PARTKEY      = PS_PARTKEY                          AND
        P_BRAND        <> 'Brand#14'                                 AND
        P_TYPE         NOT LIKE 'PROMO BRUSHED%'
        AND
        P_SIZE         IN (42, 34, 11, 27, 28, 38, 32, 3)    AND
        PS_SUPPKEY     NOT IN (       SELECT  S_SUPPKEY
                                      FROM    SUPPLIER
                                      WHERE   S_COMMENT      LIKE
'%Customer%Complaints%'
                                )
GROUP   BY      P_BRAND,
                P_TYPE,
                P_SIZE
ORDER   BY      SUPPLIER_CNT   DESC,
                P_BRAND,
                P_TYPE,
                P_SIZE



OUTPUT OF EXECUTE STREAM 0 QUERY 16 (MAX 200 ROWS)
-------------------------------------------------------------

P_BRAND    P_TYPE                    P_SIZE      SUPPLIER_CNT
---------- ------------------------- ----------- ------------
Brand#34   ECONOMY PLATED TIN        38          13527
Brand#53   STANDARD BURNISHED STEEL  27          13510
Brand#25   STANDARD BRUSHED STEEL    32          13503
Brand#53   STANDARD PLATED TIN       3           13489
Brand#45   MEDIUM BURNISHED STEEL    28          13482
Brand#31   STANDARD BURNISHED TIN    27          13481
Brand#44   ECONOMY PLATED NICKEL     32          13480
Brand#33   LARGE BURNISHED NICKEL    11          13466
Brand#24   STANDARD BURNISHED NICKEL 42          13457
Brand#34   PROMO POLISHED STEEL      42          13454
Brand#51   PROMO ANODIZED COPPER     38          13454
Brand#21   STANDARD BURNISHED STEEL  27          13444
Brand#51   PROMO ANODIZED TIN        42          13435
Brand#34   STANDARD PLATED TIN       38          13426
Brand#12   LARGE BRUSHED STEEL       11          13424
Brand#22   STANDARD POLISHED STEEL   42          13420
Brand#32   SMALL POLISHED BRASS      42          13418
Brand#35   STANDARD ANODIZED TIN     27          13409
```

Brand#55	ECONOMY PLATED NICKEL	28	13408
Brand#23	LARGE ANODIZED COPPER	34	13405
Brand#25	MEDIUM BRUSHED COPPER	32	13403
Brand#25	STANDARD BURNISHED BRASS	32	13400
Brand#11	STANDARD BRUSHED TIN	38	13398
Brand#44	LARGE ANODIZED NICKEL	3	13397
Brand#45	LARGE POLISHED COPPER	42	13394
Brand#23	ECONOMY BRUSHED TIN	34	13389
Brand#25	SMALL ANODIZED BRASS	28	13389
Brand#55	PROMO BURNISHED STEEL	32	13389
Brand#11	PROMO POLISHED STEEL	28	13387
Brand#21	SMALL PLATED COPPER	3	13385
Brand#54	MEDIUM ANODIZED STEEL	34	13385
Brand#54	STANDARD POLISHED NICKEL	42	13385
Brand#44	STANDARD ANODIZED STEEL	42	13382
Brand#21	LARGE POLISHED BRASS	28	13381
Brand#34	SMALL PLATED COPPER	38	13380
Brand#21	STANDARD PLATED NICKEL	42	13379
Brand#34	SMALL BRUSHED COPPER	42	13378
Brand#15	SMALL BRUSHED BRASS	11	13372
Brand#34	ECONOMY BURNISHED STEEL	3	13370
Brand#13	ECONOMY ANODIZED COPPER	28	13369
Brand#33	PROMO BURNISHED TIN	32	13366
Brand#42	PROMO ANODIZED STEEL	42	13365
Brand#43	PROMO ANODIZED STEEL	3	13365
Brand#11	SMALL PLATED STEEL	34	13363
Brand#12	SMALL PLATED STEEL	28	13359
Brand#11	PROMO ANODIZED TIN	42	13355
Brand#52	LARGE ANODIZED STEEL	3	13355
Brand#13	ECONOMY BURNISHED STEEL	38	13353
Brand#23	STANDARD POLISHED STEEL	11	13353
Brand#32	SMALL BURNISHED STEEL	34	13351
Brand#54	STANDARD BRUSHED NICKEL	27	13350
Brand#11	ECONOMY BURNISHED NICKEL	32	13349
Brand#24	PROMO POLISHED BRASS	28	13347
Brand#25	LARGE ANODIZED TIN	27	13345
Brand#24	SMALL PLATED NICKEL	28	13344
Brand#44	STANDARD BRUSHED NICKEL	32	13343
Brand#52	PROMO ANODIZED NICKEL	28	13342
Brand#53	PROMO POLISHED COPPER	11	13342
Brand#12	ECONOMY ANODIZED STEEL	42	13341
Brand#11	LARGE ANODIZED TIN	27	13340
Brand#31	MEDIUM BURNISHED NICKEL	32	13339
Brand#44	SMALL PLATED TIN	34	13339
Brand#33	SMALL PLATED COPPER	27	13338
Brand#25	ECONOMY PLATED STEEL	32	13337
Brand#43	LARGE BURNISHED BRASS	32	13334
Brand#55	ECONOMY ANODIZED NICKEL	3	13333
Brand#22	PROMO POLISHED BRASS	32	13331
Brand#24	ECONOMY BURNISHED STEEL	38	13331
Brand#25	SMALL PLATED STEEL	42	13331
Brand#51	SMALL BRUSHED STEEL	38	13331
Brand#32	STANDARD POLISHED BRASS	32	13330
Brand#11	MEDIUM ANODIZED BRASS	32	13329
Brand#42	SMALL POLISHED NICKEL	11	13329
Brand#53	ECONOMY POLISHED COPPER	11	13329
Brand#33	PROMO POLISHED STEEL	27	13323
Brand#34	MEDIUM PLATED STEEL	32	13323
Brand#43	ECONOMY ANODIZED TIN	28	13323
Brand#21	STANDARD BRUSHED NICKEL	42	13322
Brand#25	STANDARD ANODIZED TIN	34	13322
Brand#33	MEDIUM PLATED COPPER	28	13322
Brand#34	MEDIUM PLATED NICKEL	11	13321
Brand#11	LARGE ANODIZED NICKEL	34	13320
Brand#44	ECONOMY PLATED TIN	27	13320
Brand#32	SMALL PLATED STEEL	27	13318
Brand#54	STANDARD PLATED STEEL	42	13318
Brand#32	SMALL POLISHED STEEL	42	13317
Brand#53	ECONOMY PLATED COPPER	38	13317
Brand#33	LARGE ANODIZED TIN	27	13316
Brand#55	LARGE PLATED NICKEL	27	13316
Brand#45	LARGE PLATED TIN	32	13315
Brand#22	LARGE POLISHED BRASS	11	13314
Brand#53	STANDARD BRUSHED NICKEL	32	13313
Brand#11	LARGE POLISHED TIN	27	13312
Brand#42	MEDIUM POLISHED COPPER	32	13312
Brand#42	PROMO POLISHED STEEL	28	13311
Brand#11	ECONOMY BURNISHED STEEL	34	13310
Brand#55	ECONOMY BURNISHED STEEL	38	13310
Brand#11	MEDIUM PLATED BRASS	32	13309
Brand#31	ECONOMY PLATED COPPER	3	13308
Brand#44	SMALL POLISHED STEEL	38	13308

[27640 more rows]

Brand#55	SMALL POLISHED NICKEL	32	12290
Brand#24	SMALL BRUSHED TIN	34	12289
Brand#23	SMALL ANODIZED NICKEL	3	12287
Brand#34	LARGE PLATED BRASS	11	12287
Brand#55	SMALL POLISHED BRASS	32	12287
Brand#42	MEDIUM POLISHED NICKEL	11	12286
Brand#21	MEDIUM POLISHED COPPER	32	12285
Brand#24	SMALL ANODIZED STEEL	38	12285
Brand#54	SMALL ANODIZED COPPER	27	12285
Brand#54	LARGE BURNISHED COPPER	28	12284
Brand#55	PROMO PLATED TIN	34	12284
Brand#33	LARGE ANODIZED BRASS	32	12283
Brand#21	SMALL BRUSHED COPPER	27	12282
Brand#15	SMALL BURNISHED STEEL	28	12281
Brand#35	ECONOMY ANODIZED STEEL	27	12281
Brand#31	MEDIUM BURNISHED STEEL	27	12279
Brand#32	LARGE ANODIZED TIN	32	12279
Brand#52	MEDIUM PLATED TIN	28	12278
Brand#23	MEDIUM BRUSHED BRASS	34	12276
Brand#51	ECONOMY BURNISHED NICKEL	32	12276
Brand#53	MEDIUM BRUSHED NICKEL	38	12276
Brand#11	MEDIUM BURNISHED TIN	27	12275
Brand#44	STANDARD BRUSHED STEEL	38	12275
Brand#33	PROMO POLISHED TIN	27	12274
Brand#15	LARGE BURNISHED COPPER	38	12273
Brand#12	PROMO POLISHED BRASS	11	12272
Brand#15	MEDIUM ANODIZED NICKEL	42	12272

```
Brand#11   STANDARD ANODIZED BRASS   42          12271
Brand#13   ECONOMY POLISHED COPPER   3           12270
Brand#21   LARGE POLISHED BRASS      38          12270
Brand#53   SMALL BURNISHED BRASS     3           12269
Brand#32   LARGE BRUSHED NICKEL      38          12268
Brand#55   MEDIUM BRUSHED NICKEL     32          12267
Brand#22   LARGE POLISHED STEEL      32          12266
Brand#23   SMALL POLISHED STEEL      11          12266
Brand#32   STANDARD BRUSHED COPPER   3           12263
Brand#22   LARGE ANODIZED STEEL      11          12262
Brand#25   ECONOMY POLISHED STEEL    28          12262
Brand#51   PROMO BURNISHED TIN       42          12262
Brand#55   ECONOMY ANODIZED NICKEL   42          12261
Brand#15   LARGE PLATED COPPER       3           12260
Brand#53   SMALL ANODIZED NICKEL     3           12260
Brand#41   MEDIUM PLATED COPPER      3           12259
Brand#44   MEDIUM ANODIZED BRASS     11          12259
Brand#23   SMALL BRUSHED NICKEL      42          12258
Brand#45   LARGE BURNISHED BRASS     11          12257
Brand#11   MEDIUM POLISHED STEEL     11          12256
Brand#45   STANDARD ANODIZED TIN     38          12255
Brand#21   SMALL ANODIZED BRASS      11          12253
Brand#43   MEDIUM ANODIZED TIN       11          12252
Brand#45   SMALL POLISHED BRASS      38          12250
Brand#55   PROMO ANODIZED BRASS      11          12250
Brand#43   LARGE PLATED COPPER       28          12248
Brand#13   MEDIUM BURNISHED STEEL    42          12246
Brand#33   MEDIUM PLATED STEEL       42          12245
Brand#53   MEDIUM ANODIZED BRASS     28          12245
Brand#11   SMALL ANODIZED COPPER     27          12243
Brand#53   LARGE ANODIZED TIN        38          12242
Brand#23   SMALL ANODIZED STEEL      27          12240
Brand#32   ECONOMY BRUSHED STEEL     27          12237
Brand#33   LARGE BRUSHED NICKEL      27          12237
Brand#23   ECONOMY PLATED NICKEL     42          12236
Brand#34   LARGE ANODIZED NICKEL     42          12234
Brand#35   SMALL ANODIZED NICKEL     3           12234
Brand#44   PROMO ANODIZED TIN        11          12231
Brand#34   ECONOMY BURNISHED COPPER  28          12227
Brand#41   SMALL ANODIZED COPPER     42          12227
Brand#34   SMALL PLATED TIN          32          12226
Brand#53   MEDIUM POLISHED STEEL     11          12225
Brand#31   ECONOMY BRUSHED STEEL     38          12223
Brand#11   STANDARD PLATED STEEL     38          12219
Brand#24   SMALL POLISHED STEEL      11          12213
Brand#13   LARGE POLISHED STEEL      3           12211
Brand#45   SMALL PLATED BRASS        38          12211
Brand#44   ECONOMY POLISHED NICKEL   3           12209
Brand#53   STANDARD BRUSHED TIN      34          12209
Brand#34   LARGE POLISHED NICKEL     28          12207
Brand#21   SMALL BRUSHED STEEL       3           12204
Brand#11   LARGE POLISHED COPPER     32          12203
Brand#45   LARGE BURNISHED BRASS     28          12202
Brand#23   LARGE ANODIZED TIN        38          12187
Brand#54   SMALL PLATED NICKEL       38          12186
Brand#13   ECONOMY ANODIZED BRASS    28          12172
Brand#13   PROMO ANODIZED NICKEL     27          12171
Brand#11   MEDIUM PLATED BRASS       27          12169
Brand#45   LARGE POLISHED COPPER     32          12169
Brand#32   MEDIUM BURNISHED COPPER   27          12168
Brand#23   LARGE ANODIZED STEEL      11          12165
Brand#51   LARGE BURNISHED BRASS     27          12151
Brand#45   SMALL BURNISHED STEEL     3           12149
Brand#42   LARGE BRUSHED COPPER      38          12145
Brand#22   SMALL PLATED TIN          38          12143
Brand#33   SMALL POLISHED COPPER     27          12132
Brand#43   SMALL BRUSHED COPPER      34          12127
Brand#11   LARGE ANODIZED NICKEL     3           12125
Brand#21   PROMO BURNISHED BRASS     28          12072
Brand#53   ECONOMY BURNISHED NICKEL  32          12069
Brand#31   PROMO ANODIZED NICKEL     34          12048
Brand#41   PROMO ANODIZED NICKEL     27          12042
Brand#53   LARGE POLISHED TIN        42          12032

(27840 row(s) affected)
```

EXECUTE STREAM 0 QUERY 17

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 17 - Small-Quantity-Order Revenue */

SELECT  SUM(L_EXTENDEDPRICE)/7.0      AS AVG_YEARLY
FROM    LINEITEM,
        PART
WHERE   P_PARTKEY      = L_PARTKEY    AND
        P_BRAND        = 'Brand#45'           AND
        P_CONTAINER    = 'WRAP BOX'           AND
        L_QUANTITY     <       (      SELECT  0.2 * AVG(L_QUANTITY)
                                      FROM    LINEITEM
                                      WHERE   L_PARTKEY      = P_PARTKEY
                                )
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 17 (MAX 200 ROWS)
------------------------------------------------------------

AVG_YEARLY
------------------------
685760261.521430

(1 row(s) affected)
```

EXECUTE STREAM 0 QUERY 18

```
------------------------------------------------------------
```

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 18 - Large Volume Customer */

SELECT  TOP 100
        C_NAME,
        C_CUSTKEY,
        O_ORDERKEY,
        O_ORDERDATE,
        O_TOTALPRICE,
        SUM(L_QUANTITY)
FROM    CUSTOMER,
        ORDERS,
        LINEITEM
WHERE   O_ORDERKEY     IN      (       SELECT  L_ORDERKEY
                                       FROM    LINEITEM
                                       GROUP   BY      L_ORDERKEY HAVING
SUM(L_QUANTITY) > 315
                               )       AND
        C_CUSTKEY      = O_CUSTKEY     AND
        O_ORDERKEY     = L_ORDERKEY
GROUP   BY      C_NAME,
                C_CUSTKEY,
                O_ORDERKEY,
                O_ORDERDATE,
                O_TOTALPRICE
ORDER   BY      O_TOTALPRICE    DESC,
                O_ORDERDATE
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 18 (MAX 200 ROWS)
------------------------------------------------------------

C_NAME                     C_CUSTKEY   O_ORDERKEY           O_ORDERDATE
O_TOTALPRICE
-------------------------- ----------- -------------------- -------------
---------- ------------------------ ------------------------
Customer#158137696         158137696   15941369219          1994-05-21
00:00:00.000 601034.190000             322.000000
Customer#158137696         158137696   7351434629           1994-05-21
00:00:00.000 601034.190000             322.000000
Customer#357399533         357399533   2882403394           1994-12-01
00:00:00.000 594606.240000             336.000000
Customer#357399533         357399533   11472337984          1994-12-01
00:00:00.000 594606.240000             336.000000
Customer#093583247         93583247    8167674722           1997-04-12
00:00:00.000 594369.660000             320.000000
Customer#093583247         93583247    16757609312          1997-04-12
00:00:00.000 594369.660000             320.000000
Customer#165172211         165172211   2217820775           1992-07-05
00:00:00.000 592840.890000             321.000000
Customer#165172211         165172211   10807755365          1992-07-05
00:00:00.000 592840.890000             321.000000
Customer#040286842         40286842    12034888871          1993-01-26
00:00:00.000 592840.890000             321.000000
Customer#040286842         40286842    3444954305           1993-01-26
00:00:00.000 592840.890000             321.000000
Customer#378195094         378195094   7126354823           1995-09-27
00:00:00.000 592840.890000             321.000000
Customer#378195094         378195094   15716289413          1995-09-27
00:00:00.000 592840.890000             321.000000
Customer#181872248         181872248   3194967172           1993-08-25
00:00:00.000 590519.630000             323.000000
Customer#181872248         181872248   11784901762          1993-08-25
00:00:00.000 590519.630000             323.000000
Customer#051653413         51653413    16132750177          1996-06-11
00:00:00.000 587841.190000             331.000000
Customer#051653413         51653413    7542815587           1996-06-11
00:00:00.000 587841.190000             331.000000
Customer#057387826         57387826    6681066983           1995-02-07
00:00:00.000 584570.860000             324.000000
Customer#057387826         57387826    15271001573          1995-02-07
00:00:00.000 584570.860000             324.000000
Customer#018687827         18687827    16873237251          1992-04-20
00:00:00.000 584553.940000             333.000000
Customer#018687827         18687827    8283302661           1992-04-20
00:00:00.000 584553.940000             333.000000
Customer#405126295         405126295   15484892897          1994-04-24
00:00:00.000 583330.300000             321.000000
Customer#405126295         405126295   6894958307           1994-04-24
00:00:00.000 583330.300000             321.000000
Customer#138145396         138145396   17546127460          1996-07-14
00:00:00.000 582883.620000             329.000000
Customer#138145396         138145396   366258304            1996-07-14
00:00:00.000 582883.620000             329.000000
Customer#138145396         138145396   8956192870           1996-07-14
00:00:00.000 582883.620000             329.000000
Customer#075972589         75972589    2279010692           1996-05-01
00:00:00.000 582316.910000             329.000000
Customer#075972589         75972589    10868945282          1996-05-01
00:00:00.000 582316.910000             329.000000
Customer#415963831         415963831   4169605479           1996-05-22
00:00:00.000 577393.810000             322.000000
Customer#415963831         415963831   12759540069          1996-05-22
00:00:00.000 577393.810000             322.000000
Customer#381102364         381102364   1715338467           1996-09-10
00:00:00.000 577393.810000             322.000000
Customer#381102364         381102364   10305273057          1996-09-10
00:00:00.000 577393.810000             322.000000
Customer#288560716         288560716   9809769988           1996-04-18
00:00:00.000 577102.450000             319.000000
Customer#288560716         288560716   1219835398           1996-04-18
00:00:00.000 577102.450000             319.000000
Customer#427662715         427662715   1190167365           1994-09-13
00:00:00.000 576382.660000             319.000000
Customer#427662715         427662715   9780101955           1994-09-13
00:00:00.000 576382.660000             319.000000
Customer#081931589         81931589    11035404516          1993-10-20
00:00:00.000 576354.780000             322.000000
```

```
Customer#081931589        81931589    2445469926          1993-10-20
00:00:00.000 576354.780000            322.000000
Customer#340500391        340500391   7354003974          1996-07-12
00:00:00.000 576354.780000            322.000000
Customer#340500391        340500391   15943938564         1996-07-12
00:00:00.000 576354.780000            322.000000
Customer#246779062        246779062   11501809863         1992-04-21
00:00:00.000 574713.160000            322.000000
Customer#246779062        246779062   2911875297          1992-04-21
00:00:00.000 574713.160000            322.000000
Customer#375797689        375797689   10274676357         1995-05-24
00:00:00.000 574713.160000            322.000000
Customer#375797689        375797689   1684741767          1995-05-24
00:00:00.000 574713.160000            322.000000
Customer#327151105        327151105   496675811           1992-12-28
00:00:00.000 574478.870000            330.000000
Customer#327151105        327151105   17676544967         1992-12-28
00:00:00.000 574478.870000            330.000000
Customer#327151105        327151105   9086610401          1992-12-28
00:00:00.000 574478.870000            330.000000
Customer#032612159        32612159    10313743907         1993-04-14
00:00:00.000 574478.870000            330.000000
Customer#032612159        32612159    1723809317          1993-04-14
00:00:00.000 574478.870000            330.000000
Customer#298828031        298828031   7627181030          1997-03-01
00:00:00.000 573376.230000            317.000000
Customer#298828031        298828031   16217115620         1997-03-01
00:00:00.000 573376.230000            317.000000
Customer#148558604        148558604   13127865957         1993-04-13
00:00:00.000 572919.390000            321.000000
Customer#148558604        148558604   4537931367          1993-04-13
00:00:00.000 572919.390000            321.000000
Customer#273209251        273209251   856530849           1994-09-02
00:00:00.000 572919.390000            321.000000
Customer#273209251        273209251   9446465415          1994-09-02
00:00:00.000 572919.390000            321.000000
Customer#099452350        99452350    15922381508         1996-05-26
00:00:00.000 572895.590000            323.000000
Customer#099452350        99452350    7332446918          1996-05-26
00:00:00.000 572895.590000            323.000000
Customer#195843005        195843005   4180994692          1996-07-16
00:00:00.000 572878.990000            322.000000
Customer#195843005        195843005   12770929282         1996-07-16
00:00:00.000 572878.990000            322.000000
Customer#165696442        165696442   9089528740          1997-12-30
00:00:00.000 572878.990000            322.000000
Customer#165696442        165696442   17679463330         1997-12-30
00:00:00.000 572878.990000            322.000000
Customer#165696442        165696442   499594150           1997-12-30
00:00:00.000 572878.990000            322.000000
Customer#001234216        1234216     9909123298          1995-08-29
00:00:00.000 571826.270000            320.000000
Customer#001234216        1234216     1319188708          1995-08-29
00:00:00.000 571826.270000            320.000000
Customer#401747870        401747870   8569460032          1994-08-24
00:00:00.000 571736.310000            320.000000
Customer#401747870        401747870   17159394598         1994-08-24
00:00:00.000 571736.310000            320.000000
Customer#159339964        159339964   17473340420         1993-09-19
00:00:00.000 571233.180000            319.000000
Customer#159339964        159339964   293471264           1993-09-19
00:00:00.000 571233.180000            319.000000
Customer#159339964        159339964   8883405830          1993-09-19
00:00:00.000 571233.180000            319.000000
Customer#344689369        344689369   5870175746          1992-11-07
00:00:00.000 570840.060000            318.000000
Customer#344689369        344689369   14460110336         1992-11-07
00:00:00.000 570840.060000            318.000000
Customer#430305292        430305292   7171406599          1992-07-23
00:00:00.000 570668.420000            318.000000
Customer#430305292        430305292   15761341189         1992-07-23
00:00:00.000 570668.420000            318.000000
Customer#418115260        418115260   12201918951         1998-07-11
00:00:00.000 570446.200000            325.000000
Customer#418115260        418115260   3611984385          1998-07-11
00:00:00.000 570446.200000            325.000000
Customer#063870244        63870244    6207185504          1992-12-04
00:00:00.000 570162.420000            330.000000
Customer#063870244        63870244    14797120070         1992-12-04
00:00:00.000 570162.420000            330.000000
Customer#015898064        15898064    13507924646         1993-06-18
00:00:00.000 570149.260000            325.000000
Customer#015898064        15898064    4917990080          1993-06-18
00:00:00.000 570149.260000            325.000000
Customer#296754410        296754410   3690856550          1997-03-25
00:00:00.000 570149.260000            325.000000
Customer#296754410        296754410   12280791140         1997-03-25
00:00:00.000 570149.260000            325.000000
Customer#162753541        162753541   14742937219         1992-10-06
00:00:00.000 569974.150000            324.000000
Customer#162753541        162753541   6153002629          1992-10-06
00:00:00.000 569974.150000            324.000000
Customer#079420369        79420369    7380136135          1995-01-15
00:00:00.000 569974.150000            324.000000
Customer#079420369        79420369    15970070725         1995-01-15
00:00:00.000 569974.150000            324.000000
Customer#140404471        140404471   1731330850          1993-01-09
00:00:00.000 569422.890000            324.000000
Customer#140404471        140404471   10321265440         1993-01-09
00:00:00.000 569422.890000            324.000000
Customer#094148504        94148504    15229799488         1996-03-21
00:00:00.000 569422.890000            324.000000
Customer#094148504        94148504    6639864898          1996-03-21
00:00:00.000 569422.890000            324.000000
Customer#248124062        248124062   12997274757         1995-03-09
00:00:00.000 568885.320000            331.000000
Customer#248124062        248124062   4407340167          1995-03-09
00:00:00.000 568885.320000            331.000000
Customer#140637170        140637170   1851977413          1993-10-12
00:00:00.000 568770.470000            335.000000
Customer#140637170        140637170   10441912003         1993-10-12
00:00:00.000 568770.470000            335.000000
```

```
Customer#046717708        46717708    4306244449          1994-02-14
00:00:00.000 568770.470000            335.000000
Customer#046717708        46717708    12896179015         1994-02-14
00:00:00.000 568770.470000            335.000000
Customer#246996451        246996451   16577579557         1994-02-24
00:00:00.000 568770.470000            335.000000
Customer#246996451        246996451   7987644967          1994-02-24
00:00:00.000 568770.470000            335.000000
Customer#411549016        411549016   13962717281         1996-07-09
00:00:00.000 568015.020000            326.000000
Customer#411549016        411549016   5372782691          1996-07-09
00:00:00.000 568015.020000            326.000000
Customer#053126278        53126278    8130580263          1992-11-20
00:00:00.000 567630.690000            316.000000
Customer#053126278        53126278    16720514853         1992-11-20
00:00:00.000 567630.690000            316.000000

(100 row(s) affected)
```

EXECUTE STREAM 0 QUERY 19

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 19 - Discounted Revenue */

SELECT  SUM(L_EXTENDEDPRICE* (1 - L_DISCOUNT))       AS REVENUE
FROM    LINEITEM,
        PART
WHERE   (       P_PARTKEY       = L_PARTKEY
                AND
                P_BRAND         = 'Brand#54'
                        AND
                P_CONTAINER     IN ( 'SM CASE', 'SM BOX', 'SM PACK', 'SM
PKG')           AND
                L_QUANTITY      >= 9
                AND
                L_QUANTITY      <= 9 + 10
                AND
                P_SIZE          BETWEEN 1 AND 5
                        AND
                L_SHIPMODE      IN ('AIR', 'AIR REG')
                AND
                L_SHIPINSTRUCT = 'DELIVER IN PERSON'
        )
        OR
        (       P_PARTKEY       = L_PARTKEY
                AND
                P_BRAND         = 'Brand#23'
                        AND
                P_CONTAINER     IN ( 'MED BAG', 'MED BOX', 'MED PKG', 'MED
PACK')  AND
                L_QUANTITY      >= 12
                AND
                L_QUANTITY      <= 12 + 10
                AND
                P_SIZE          BETWEEN 1 AND 10
                AND
                L_SHIPMODE      IN ('AIR', 'AIR REG')
                AND
                L_SHIPINSTRUCT = 'DELIVER IN PERSON'
        )
        OR
        (       P_PARTKEY       = L_PARTKEY
                AND
                P_BRAND         = 'Brand#51'
                        AND
                P_CONTAINER     IN ( 'LG CASE', 'LG BOX', 'LG PACK', 'LG
PKG')           AND
                L_QUANTITY      >= 27
                AND
                L_QUANTITY      <= 27 + 10
                AND
                P_SIZE          BETWEEN 1 AND 15
                AND
                L_SHIPMODE      IN ('AIR', 'AIR REG')
                AND
                L_SHIPINSTRUCT = 'DELIVER IN PERSON'
        )

OUTPUT OF EXECUTE STREAM 0 QUERY 19 (MAX 200 ROWS)

REVENUE
-------------------------
11739209274.903288

(1 row(s) affected)
```

EXECUTE STREAM 0 QUERY 20

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 20 - Potential Part Promotion */

SELECT  S_NAME,
        S_ADDRESS
FROM    SUPPLIER,
        NATION
WHERE   S_SUPPKEY       IN      (       SELECT  PS_SUPPKEY
                                        FROM    PARTSUPP
                                        WHERE   PS_PARTKEY in  (
        SELECT  P_PARTKEY
```

```
        PART
        P_NAME like 'papaya%'
                                                          )
                                    PS_AVAILQTY    >      (
        SELECT  0.5 * sum(L_QUANTITY)
        LINEITEM
        L_PARTKEY      = PS_PARTKEY   AND
        L_SUPPKEY      = PS_SUPPKEY   AND
        L_SHIPDATE     >= '1994-01-01'              AND
        L_SHIPDATE     < dateadd(yy,1,'1994-01-01')
                                                          )
                               )       AND
        S_NATIONKEY    = N_NATIONKEY  AND
        N_NAME         = 'KENYA'
ORDER   BY      S_NAME
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 20 (MAX 200 ROWS)
------------------------------------------------------------
```

S_NAME	S_ADDRESS
Supplier#000000076	JBhSBa3cLYvNgHUYtUHmtECCD
Supplier#000000079	p0u3tztSXUD2J8vFfLNFNKsrRRv7qyUtTBTA
Supplier#000000083	WRJUkzCn050seVz57oAfrbCuw
Supplier#000000165	iPso5qCxSnxaNsRe9AU05Vl9hWm5oHIS
Supplier#000000229	ycjgLrk,w8DcakfwTS1SO5kVch
Supplier#000000307	3wL9YHFIvddxzh3mwy6SSrpfmzKvwAGmXK
Supplier#000000480	q8,LH5UQiP3Tv60slOsFzX,HM0JPcwM0rD7eg d
Supplier#000000717	hhUrgvyxsdTfzGY4OrQSHeZmMNB2L75xk
Supplier#000001022	ERUtHzpZ3GXDESNGi0o72 FP1AAslki
Supplier#000001050	Eu5ETUhMtI
Supplier#000001204	fROOqRgBPrHagHhP9A9XvGMWSm9iONll
Supplier#000001364	hZloe XgNqB5f06b74VCiDRd1bea5tMvc,Q1ACF
Supplier#000001369	NWOVSURnsQ5tfYI2CsAItodX4dG
Supplier#000001772	ueuEYuzE0mR1uw4zeNIFG
Supplier#000001893	dNN p6AskdHyftPQQrsG87kjT6Rq6owcLOYIvw
Supplier#000001964	5OlSBcKlYeqgnneJNh
Supplier#000002477	FVLqfD4TzY5s4qoxgB3BUkVOVOR1qmtNptdH
Supplier#000002748	67Y1bH8enJlO81k QtJRWIst
Supplier#000002813	9 FfkJtSNtSks3
Supplier#000002900	G0 wchKDgWmciVdmWz,4nlOOfR FIHmUL7c,
Supplier#000002991	NC0h3TZcVC
Supplier#000003149	axCWHGuM6s HGRQmQ5dSfXZRJuB,aj
Supplier#000003285	rDe8Y 4pVW 1YdzBmyk
Supplier#000003320	0uizitM07M8oV
Supplier#000003425	qpPzLbrWkeO,q1qbsBoG7vIm9aS23G7Tx H
Supplier#000003496	Pi4sY9dBGlns
Supplier#000003889	xEsTSa5oG8BjL0SJdEdCcTg2LhOguyvQ
Supplier#000004139	RPGuceC07eTzSMinFHQA35A
Supplier#000004354	ZucRIz6o8ODOsMs75T3Z,E3XGfR
Supplier#000004382	oB1Kdw13dTxFZkFmg
Supplier#000004696	J1qrxkoTf hOSu64XOflrzugGVsC7PGCl9X0
Supplier#000004726	m8N5E7nFKIx6ot,I,k W2Ry51a6p1nJksAJko4M
Supplier#000004963	F5naCEAdQhhjm3IwJsla7 OF6mMbbjtH90
Supplier#000005121	tSJ3v6tz8dkFGm1nBLzG5TDKWlwunrmsTbN
Supplier#000005281	bWv9QtKGuhX5SLgnGPjfQYyJuKxnwhhuq2bq
Supplier#000005679	5gSa,3ctOY2w4RjlfPnsIa n
Supplier#000005996	tfOVqjUrB7TTxCZ,OsPZZynV,C7wYT21HYu9I,Mp
Supplier#000006191	lYf3sL8KjpAyv,dFT
Supplier#000006668	c3En1x00tCL
Supplier#000006702	9AhU1MrHSCnKJQio l5sEd0zYF1
Supplier#000006734	vmaZ8HNigX9UX
Supplier#000006754	kMUCGph4PWmLLdBxHBZCypMzEoZ 1MVMSkl
Supplier#000007289	GoYBOvCn2XuN,7i
Supplier#000007354	aeK 25BHHY2z1O 7CjrjQ32WCBuTp,XbFZAc4 TE
Supplier#000007717	jXkaR6a 1imfl
Supplier#000007780	Vt6L80UlNf6cFi
Supplier#000007947	zWZTLm57m4Jm8uaes4
Supplier#000008297	0l2KCe1ocMaACSDEQdWAwH0K
Supplier#000008412	gOBtQQlqTMmPpdlkyYRNfnYp,Dvu
Supplier#000008655	h78,rbPDGunL
Supplier#000008728	akbOO5Qpw4HRYupOdz5jXK
Supplier#000009101	Up6zowOQBEhC WmO90
Supplier#000009240	xNRyBB09ybecSGN
Supplier#000009297	l8xTeKqe4sDvbBpiKqkUhiRUw JQu
Supplier#000009299	UKUpKpXccW5X4r1n1UIIxpjpJH
Supplier#000009362	ReNJhH k9lB3GU2PNVpSLwOu
Supplier#000009706	VHshTcwi8JFa2O2
Supplier#000009763	5014E0NVyt1ktVX,8MCrtaGpDhl
Supplier#000009768	,1vdQhM1gDmOI09jy2XqgsUYcL1BSiG7
Supplier#000010654	1Q jTQ9uHsT
Supplier#000010683	8bK6iJ3heCdNBxjH
Supplier#000010866	AXEF5AtvI4Cycx6RUi
Supplier#000011010	gGxAnRzzJ,3RnXqP1Mp M2gqFAA5GjkUb8o6uPGd
Supplier#000011037	5KWL5Ai6nQrpR
Supplier#000011365	v,ooaUS5bbnImOd2cwg
Supplier#000011648	fCyb47mEJN3IHGSuqCDPGxl6m
Supplier#000011800	SlaMTHfNiRBkO8,
Supplier#000012097	izMvMeJCdftV
Supplier#000012113	bsBqbxtH RNMVnucUzc
Supplier#000012235	kyjzgRgmvCDy
Supplier#000012256	6piKkVtV6e,32 Uc8 pALgxus5xNL
Supplier#000012288	mqe09ij 8q,a0Q,Ky20T
Supplier#000012744	A3aIv0 JGbodHg8ZzD9n,qXVysnfhAW,gqir11
Supplier#000012747	Jk9Xzu2vP72x,2DNvTQEJiv75WMD
Supplier#000012938	FmXkoduwQz9JqyTcJanmBTz0u8VvfAHgm49PuPM
Supplier#000012941	tf N73iibLoM0Jzkozc
Supplier#000013141	h0xj2vJ80jANAO8kx39 3iwW5pnvqdtZQy
Supplier#000013214	FncjbbCkltBNUnaRCh9bbBNNLdL35FYTBsmW5QD
Supplier#000013656	dKy4KdspZhdTUapYajAZhmx
Supplier#000013956	h9dSwccUs1HvGJayJ
Supplier#000013990	nQyDu BEjEp9diCfeq

```
Supplier#000014369      tWG09 86gjQ6KVCUmEDM
Supplier#000014512      VLCPlPTqRItVXeJfMpuoL5d1ofJvk 4
Supplier#000014589      7b0PSuaApULZvJHq4lgC
Supplier#000014638      9nqoVFILNw
Supplier#000014754      8de7zkMlzYTN6F
Supplier#000014773      lnkPWW,SfDWFLVw
Supplier#000014898      OswMBlxE0jwQInfK2W5t2Y9kZuhT1vmmLwBPO
Supplier#000014967      XLwIimXfSIdgKW7Vi6RoJEbKBVSQsL9YE7BphR
Supplier#000014970      hK5o2EnWa4XhUHhA4k
Supplier#000015373      frFS,T,lBNEKTqOTq GpxpcE0
Supplier#000015462      OgfoDkd1gQ9EeI1HVjwcbfneUw0f
Supplier#000015804      y2YseM5wxlJK8NgbTEmuo8rffBNNtrde
Supplier#000015972      HfQRp tEoeaOgLFdoCbD4d4i2Qm0emivC,S
Supplier#000015995      hQLg7ZeN73AWPoMAj9PJ,NMS,9Z3Rgs9LcA5xY
Supplier#000016127      sjwPw6uffXYMCC6a
Supplier#000016227      5xsOHG0N5VNccg4UzJOSXXi8
Supplier#000016354      TC9cyGy3O 6f3Cr0BK
Supplier#000016629      YwDF1jOzQyt3
Supplier#000016793      Ux5sP1D6LWVlRrlJnKr4Q

[201098 more rows]

Supplier#029986290      KDvlnWSZBdhXb9Tg4,uPQFN3QzFaKRlVFpsM
Supplier#029986455      VnXp6U6BY6rFnypg
Supplier#029986497      JtY2fjO3ri5Ai30S054wIBx0
Supplier#029986558      2MuTaXjqnROtE
Supplier#029986567      6,5o6saVbxR,9nZzVs2OLyiS8QRvB
Supplier#029986596      mipFb2 umFPs7PPs1t
Supplier#029986696      jiplJmnjzPDSkNy9Fo6e7
Supplier#029986753      N2Zjxem9a3qRAstYuMn,UzS3Xn2LOru
Supplier#029987023      12TjyInovh1bST7cYEwAZz2Fy
Supplier#029987383      EvU 6TewR3KowauWl
Supplier#029987919      GcI2lMknGUU2
Supplier#029988091      BYrkLHy9W6JrN6n9UbBSYl8YcaqUp9N40OKf
Supplier#029988227      BHuT5DPc5OAmd
Supplier#029988743      pd8wvhqMbJxZAslxM0U7Iq7VcjO
Supplier#029988750      C9xi C v,VwO
Supplier#029988820      ayzI4uRsenlEwVjnK3jFch7WNwbNM1zYALTb
Supplier#029988830      ,snVKp3X 8yDsm
Supplier#029988864      iuQN,g5xTZaCCSGO9r3ri DfwQMeB
Supplier#029988908      Iv2NMh5jIbN4
Supplier#029989033      AUsc1pvgUH3SypLyhzh4UUc6tQsOB7
Supplier#029989190      3ZpucQoMQ4D9GX4fJFCp7jg
Supplier#029989318      Rv7dM ,VGGZp7hr8qMKG2Z0oe7muB4
Supplier#029989497      EEvl9RtJFGUysgxGCdTAkkF0
Supplier#029989512      RODy,GA4bxFXYxy9Y2
Supplier#029989530      ROjWZJCjf9pIvEiUv56fqdTd7nN6 dgbfN
Supplier#029989533      qLi5ksuy0VNZBtzDZICPKvmrF50ALP0aS41C1vL
Supplier#029989788      KLQYqGVfMA6UQaDuyxeS,uw97MYBMhJHyKVgp
Supplier#029989923      w28Lujt7fLX4sKhG9jR0mRcvOs
Supplier#029989925      SiVG83THp3d
Supplier#029990192      KMWCXJB13reJtqSWckO6C bZd9nU
Supplier#029990261      vLbl QwB2SDDPfcVNk7fLQ0BcOBxBiEVf33YCJ8
Supplier#029990508      CZvjvHLMNU6BDE5ERCzTJUY
Supplier#029990667      pZZwCIxKU6iWU
Supplier#029990722      EGsHU4f8w7fHCi6ttdQ3XGIxa DE rVSnMQ48oP
Supplier#029990818      LVdOmKZYb iHy13TIoY9huQAnaHrYgBlCNPSjl
Supplier#029990820      lw45YP 5yZod4BEGGjrb3VIdL96 iyyD75UyF
Supplier#029991403       RVHeO2djRcbp6Ifi
Supplier#029991643      ,TcAQJf6sXrJ7HRnxKwX kvvVmGeqz
Supplier#029991648      mpghhz0JDazRdabWPIWCA0 ,NTpj9iLvH69D
Supplier#029991718      PsgXd0QX3b
Supplier#029991740      DlN8zSX9rolZbi3f0r6tzrpt
Supplier#029991821      qLr77973aM3J7S5gr9ce wh,
Supplier#029992060      YbH6SWIRYYNslHZObU
Supplier#029992337      KRiEWq xmp ou9FPA1auYMICqvYnCOvZi
Supplier#029992566      cumNNV2fG4Ej2XQ3sM1EqAUtg2FaYrZAwU
Supplier#029992763      JpgVvdYgEWHkRzhzfMetIABAhVEOTEHd,
Supplier#029992836      jefc5DzKMgwXN,RK
Supplier#029992849      vfxUZlgC8OqwLXqR,cB90bevfb6
Supplier#029992892      XucNqVedHO6857u9qf
Supplier#029992901      Gpbc9tSZ5t4pj0eT2p Mev,H
Supplier#029993015      UjbTQG5iyOl2dXKSSt3GguQproInEtRK9hrz2
Supplier#029993019      509hS9fOiRiIad3XhQRRPgUxU7SG4POR
Supplier#029993092      Y2GFPD6ieNPi7P4hs
Supplier#029993129      iF35QA6k43fu7WxsSjCGgL2iwoQ9hyEPuI6
Supplier#029993175      EFouNgpAv9D56l51jIMB PpJ7vBT
Supplier#029993184      tGCtHB 3vm,xYyQYhCcLt6pv0o
Supplier#029993187      72XHi2sa9nqY5kCV0ZDdKksQ1O,4xUNsdCtW70Y
Supplier#029993261      4cbtHm6cAx0YkcIr ui0LGHpP
Supplier#029993642      G5gIx4x0sNk
Supplier#029993727      pKSdKwDWDzlM74ig
Supplier#029993740      6IjvngHzKuemR5rPF33t,Qu
Supplier#029994240      vO7JdLFv3blOfM2ciJ
Supplier#029994405      nY5NX8pNV6ubw7TP
Supplier#029994443      vvJF8okcncH0Uz
Supplier#029994615      FkW5qygFtaAvn4O1jAQ,iG lgcy
Supplier#029994719      a0YRbQjLdcLlz9LK
Supplier#029994756      ZLk64BG3So
Supplier#029994884      vPt7a37ZCJUYEC
Supplier#029994917      T0mVUmyYF8Ba26a iFyRzoLf8EpVFxMvyecEi25Z
Supplier#029994919      dJSwgwR49D s3pCaXZn 8F h
Supplier#029994982      7O1dGC8kjFY9ucRMq535CY
Supplier#029995055      FghFrKSEqU9aGE6QbN4Flu,meFOMRSB6N
Supplier#029995171      LzdAWU5AbED9GU
Supplier#029995230      TlAfX8qhnwGBws,Mx9RnvT00dhHKCZfuftSkw
Supplier#029995419      znjChS59fDMYwsw8
Supplier#029995557      KfKeunVbpL0ea7BUHxcyBKiB
Supplier#029995790      BXsSWHmHivqRRYAsL8f
Supplier#029995895      OAvmb58Y10
Supplier#029996026      NTb4MyLEd2ZM5hfwE
Supplier#029996033      eVtiWTDzHJlz03S
Supplier#029996166      WlSdvIQs0Nrsolk pc6WUNc
Supplier#029996336      5iXIAQAHM3L50QKNrzF7BLg,SrS6a6O
Supplier#029996558      YWtxOiODXpCFKZ2iCqnGmVZdic684m,
Supplier#029996603      x, ctJGy6i r3ZOhNVRHOzz7afP9b6xoNZqSN
Supplier#029996732      WTVmSfXU2 vaRYRn0WqGOIqYFW0
Supplier#029996843      Yy3aZqYfE2GQ2uu7Cp
Supplier#029996879      lMFB50aZEI4tM1Ekm6
Supplier#029996953      8 sfNmhgmRCkV
Supplier#029997174      YVOqLV5XTgIxMHows62m7ZyF15tw5czxNdVav
Supplier#029997238      QELmrPeQ2RhA4Ex5llN
```

```
Supplier#029997911        2P wCG3pzrs6i9G,PlLihH9PkMb6tb45V26zB5
Supplier#029998123        mH9adOpoS rDtEbLru Qz17PYR9Cw,V2aad3A
Supplier#029998151        jlqiL3dc1OULOGY,3HBWocaY
Supplier#029998233        u2G9Q2Zk5,3VG5gf4ld5y69nK
Supplier#029998375        byElcr99e9Mt5
Supplier#029998423        aRn HXgcN3JYKNMUtTh6lIK32rz AAZd
Supplier#029998447        wzp6d50TwboQqiBAUO4KyYipobnp2rAVgiitE3o
Supplier#029999117        Jy,u4DwN0C8Q
Supplier#029999534        rWMdt6DFNdFo5oPtybJwdF1o,mSKrZUGHRDgmUg
Supplier#029999833        Oj4E1RNBF7mfLM5Fv9Oc A

(201298 row(s) affected)
```

EXECUTE STREAM 0 QUERY 21

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 21 - Suppliers Who Kept Orders Waiting */

SELECT  TOP 100
        S_NAME,
        COUNT(*)        AS NUMWAIT
FROM    SUPPLIER,
        LINEITEM L1,
        ORDERS,
        NATION
WHERE   S_SUPPKEY               = L1.L_SUPPKEY          AND
        O_ORDERKEY              = L1.L_ORDERKEY                 AND
        O_ORDERSTATUS           = 'F'                   AND
        L1.L_RECEIPTDATE        > L1.L_COMMITDATE       AND
        EXISTS  (       SELECT  *
                        FROM    LINEITEM L2
                        WHERE   L2.L_ORDERKEY   = L1.L_ORDERKEY         AND
                                L2.L_SUPPKEY    <> L1.L_SUPPKEY
                )       AND
        NOT EXISTS      (       SELECT  *
                                FROM    LINEITEM L3
                                WHERE   L3.L_ORDERKEY           =
L1.L_ORDERKEY           AND
                                        L3.L_SUPPKEY            <>
L1.L_SUPPKEY            AND
                                        L3.L_RECEIPTDATE        >
L3.L_COMMITDATE
                        )       AND
        S_NATIONKEY     = N_NATIONKEY   AND
        N_NAME          = 'ETHIOPIA'
GROUP   BY      S_NAME
ORDER   BY      NUMWAIT DESC,
                S_NAME
```

```
OUTPUT OF EXECUTE STREAM 0 QUERY 21 (MAX 200 ROWS)
------------------------------------------------------------

S_NAME                    NUMWAIT
------------------------- -----------
Supplier#000106125        57
Supplier#013830748        54
Supplier#027276538        54
Supplier#018573267        52
Supplier#021143098        52
Supplier#000644785        49
Supplier#004607985        49
Supplier#006045808        49
Supplier#007976533        49
Supplier#016536410        49
Supplier#018908726        49
Supplier#021561344        49
Supplier#025342947        49
Supplier#010223703        48
Supplier#014806978        48
Supplier#020477143        48
Supplier#026728840        48
Supplier#027827424        48
Supplier#028391554        48
Supplier#000739292        47
Supplier#002428522        47
Supplier#010052582        47
Supplier#018509427        47
Supplier#020479808        47
Supplier#020723490        47
Supplier#020879723        47
Supplier#023549296        47
Supplier#029603724        47
Supplier#000242841        46
Supplier#000754375        46
Supplier#000951661        46
Supplier#002410982        46
Supplier#004416753        46
Supplier#011712037        46
Supplier#011714774        46
Supplier#015109192        46
Supplier#022271187        46
Supplier#022736657        46
Supplier#023157278        46
Supplier#027749592        46
Supplier#002261726        45
Supplier#004644081        45
Supplier#005255872        45
Supplier#007784654        45
Supplier#010180506        45
Supplier#010324569        45
Supplier#015431440        45
Supplier#018220405        45
Supplier#020120103        45
Supplier#021373846        45
Supplier#022109116        45
```

```
Supplier#022995942        45
Supplier#024911033        45
Supplier#027477043        45
Supplier#028430179        45
Supplier#029946397        45
Supplier#001288535        44
Supplier#001578479        44
Supplier#002176063        44
Supplier#002655211        44
Supplier#003638726        44
Supplier#003870743        44
Supplier#003902458        44
Supplier#004208848        44
Supplier#005765010        44
Supplier#006225779        44
Supplier#006271796        44
Supplier#009334337        44
Supplier#009439728        44
Supplier#010156609        44
Supplier#011530456        44
Supplier#011585908        44
Supplier#013058487        44
Supplier#015535003        44
Supplier#019237953        44
Supplier#019929437        44
Supplier#020438877        44
Supplier#020476514        44
Supplier#021125520        44
Supplier#021174325        44
Supplier#021527714        44
Supplier#022891606        44
Supplier#023872335        44
Supplier#025908053        44
Supplier#025934330        44
Supplier#026635965        44
Supplier#026782299        44
Supplier#027206146        44
Supplier#028388140        44
Supplier#000217633        43
Supplier#000533401        43
Supplier#001176836        43
Supplier#001241746        43
Supplier#001895881        43
Supplier#002153239        43
Supplier#002224270        43
Supplier#002246432        43
Supplier#002724429        43
Supplier#004746001        43
Supplier#005279621        43

(100 row(s) affected)
```

EXECUTE STREAM 0 QUERY 22

```
-- using 1026004625 as a seed to the RNG

/* TPC_H  Query 22 - Global Sales Opportunity */

SELECT  CNTRYCODE,
        COUNT(*)        AS NUMCUST,
        SUM(C_ACCTBAL) AS TOTACCTBAL
FROM    (       SELECT  SUBSTRING(C_PHONE,1,2) AS CNTRYCODE,
                        C_ACCTBAL
                FROM    CUSTOMER
                WHERE   SUBSTRING(C_PHONE,1,2) IN      ('24', '29', '19',
'12', '28', '27', '11')       AND
                        C_ACCTBAL               >       (       SELECT
        AVG(C_ACCTBAL)
                                                                FROM
        CUSTOMER
                                                                WHERE
        C_ACCTBAL       > 0.00  AND

        SUBSTRING(C_PHONE,1,2) IN     ('24', '29', '19', '12', '28', '27',
'11')
                                                                )       AND
                        NOT EXISTS      (       SELECT  *
                                                FROM    ORDERS
                                                WHERE   O_CUSTKEY       =
C_CUSTKEY
                                        )
        )       AS CUSTSALE
GROUP   BY      CNTRYCODE
ORDER   BY      CNTRYCODE



OUTPUT OF EXECUTE STREAM 0 QUERY 22 (MAX 200 ROWS)
------------------------------------------------------------

CNTRYCODE NUMCUST     TOTACCTBAL
--------- ----------- ------------------------
11        2725734     20445703244.369995
12        2727995     20464459392.869980
19        2727449     20448145355.459991
24        2726019     20447738804.169975
27        2724844     20436882217.630009
28        2724953     20437661796.009995
29        2727912     20460115948.220024

(7 row(s) affected)
```

Appendix D : Seed & Query substitution

STREAM 00 SEED 1026004625

```
14      1997-02-01
2       13      NICKEL  EUROPE
9       cornflower
20      papaya  1994-01-01      KENYA
6       1997-01-01      0.04    24
17      Brand#45        WRAP BOX
18      315
8       CANADA  AMERICA SMALL ANODIZED STEEL
21      ETHIOPIA
13      pending deposits
3       HOUSEHOLD       1995-03-20
22      24      29      19      12      28      27      11
16      Brand#14        PROMO BRUSHED   42      34      11      27      28
        38      32      3
4       1996-02-01
11      ARGENTINA       0.0000001000
15      1994-03-01
1       120
10      1993-06-01
19      Brand#54        Brand#23        Brand#51        9       12      27
5       MIDDLE EAST     1997-01-01
7       FRANCE  CANADA
12      RAIL    FOB     1994-01-01
```

STREAM 01 SEED 1026004626

```
21      RUSSIA
3       AUTOMOBILE      1995-03-06
18      313
5       AFRICA  1997-01-01
11      KENYA   0.0000001000
7       UNITED KINGDOM SAUDI ARABIA
6       1997-01-01      0.09    25
20      blanched        1997-01-01      EGYPT
17      Brand#42        WRAP PACK
12      AIR     RAIL    1997-01-01
16      Brand#54        MEDIUM ANODIZED         46      14      45      16
        20      5       26      27
15      1996-09-01
13      pending deposits
10      1994-04-01
2       1       TIN     AFRICA
8       SAUDI ARABIA    MIDDLE EAST     STANDARD POLISHED STEEL
14      1997-06-01
19      Brand#51        Brand#51        Brand#41        4       13      23
9       burlywood
22      12      29      13      11      18      30      23
1       67
4       1993-11-01
```

STREAM 02 SEED 1026004627

```
6       1997-01-01      0.06    24
17      Brand#44        WRAP CAN
14      1997-09-01
16      Brand#34        ECONOMY PLATED 9        26      13      39      17
        44      23      15
19      Brand#53        Brand#34        Brand#45        10      14      20
10      1995-01-01
9       bisque
2       38      COPPER  EUROPE
15      1994-06-01
8       JAPAN   ASIA    STANDARD BURNISHED STEEL
5       AMERICA 1997-01-01
22      15      28      26      18      20      30      10
12      SHIP    RAIL    1997-01-01
7       MOROCCO JAPAN
13      pending deposits
18      314
1       75
4       1996-06-01
20      linen   1995-01-01      ROMANIA
3       FURNITURE       1995-03-22
11      BRAZIL  0.0000001000
21      KENYA
```

STREAM 03 SEED 1026004628

```
8       EGYPT   MIDDLE EAST     PROMO BRUSHED COPPER
5       ASIA    1997-01-01
4       1994-03-01
6       1997-01-01      0.04    24
17      Brand#41        SM BOX
7       GERMANY EGYPT
1       83
18      312
22      17      15      28      13      22      31      25
14      1997-12-01
9       yellow
10      1993-10-01
15      1997-01-01
11      MOROCCO 0.0000001000
20      tan     1994-01-01      INDIA
2       26      BRASS   AMERICA
21      FRANCE
19      Brand#15        Brand#22        Brand#44        5       15      27
13      pending deposits
16      Brand#14        STANDARD POLISHED       30      46      28      20
        44      31      21      27
12      FOB     RAIL    1997-01-01
3       MACHINERY       1995-03-08
```

STREAM 04 SEED 1026004629

```
5       EUROPE  1993-01-01
21      UNITED KINGDOM
14      1993-03-01
19      Brand#12        Brand#55        Brand#33        10      16      23
15      1994-09-01
17      Brand#43        SM PACK
12      MAIL    TRUCK   1993-01-01
```

```
6       1993-01-01      0.09    25
4       1996-10-01
9       thistle
8       VIETNAM ASIA    PROMO PLATED COPPER
16      Brand#54        LARGE ANODIZED 10       17      38      14      9
        29      18      25
11      CANADA  0.0000001000
2       14      NICKEL  EUROPE
10      1994-07-01
18      314
1       91
13      pending packages
7       UNITED STATES   VIETNAM
22      14      26      11      12      28      17      29
3       FURNITURE       1995-03-24
20      gainsboro       1997-01-01      UNITED KINGDOM

STREAM 05 SEED 1026004630
21      MOROCCO
15      1997-04-01
4       1994-07-01
6       1993-01-01      0.07    25
7       MOZAMBIQUE      JORDAN
16      Brand#34        PROMO BURNISHED         9       38      32      20
        46      29      23      19
19      Brand#14        Brand#43        Brand#33        5       17      30
18      315
14      1993-06-01
22      23      13      21      11      27      32      15
11      MOROCCO 0.0000001000
13      pending packages
3       MACHINERY       1995-03-10
1       99
2       2       TIN     AMERICA
5       MIDDLE EAST     1993-01-01
8       JORDAN  MIDDLE EAST     PROMO ANODIZED COPPER
20      red     1996-01-01      JORDAN
12      TRUCK   REG AIR 1997-01-01
17      Brand#45        SM CAN
10      1993-04-01
9       slate

STREAM 06 SEED 1026004631
10      1994-02-01
3       BUILDING        1995-03-26
15      1995-01-01
13      unusual packages
6       1993-01-01      0.04    24
8       ETHIOPIA        AFRICA  ECONOMY POLISHED COPPER
9       saddle
7       INDIA   ETHIOPIA
4       1997-02-01
11      CANADA  0.0000001000
22      10      27      17      13      30      25      15
18      313
12      RAIL    TRUCK   1993-01-01
1       107
5       AFRICA  1993-01-01
16      Brand#14        SMALL POLISHED 4        38      12      6       22
        11      13      41
2       40      COPPER  MIDDLE EAST
14      1993-10-01
19      Brand#21        Brand#21        Brand#32        1       18      26
20      chocolate       1994-01-01      CANADA
17      Brand#42        LG BOX
21      INDIA

STREAM 07 SEED 1026004632
18      314
8       RUSSIA  EUROPE  ECONOMY BURNISHED COPPER
20      moccasin        1993-01-01      PERU
21      ALGERIA
2       27      STEEL   AMERICA
4       1994-11-01
22      24      26      22      23      20      31      25
17      Brand#44        LG PACK
1       115
11      MOZAMBIQUE      0.0000001000
9       puff
19      Brand#23        Brand#14        Brand#21        6       19      22
3       MACHINERY       1995-03-12
13      unusual packages
5       AMERICA 1993-01-01
7       ALGERIA RUSSIA
10      1994-11-01
16      Brand#54        LARGE BRUSHED   3       18      22      36      23
        1       27      45
6       1993-01-01      0.02    25
14      1994-01-01
15      1997-08-01
12      REG AIR TRUCK   1993-01-01

STREAM 08 SEED 1026004633
19      Brand#21        Brand#42        Brand#25        1       20      30
1       63
15      1995-04-01
17      Brand#41        LG CAN
5       EUROPE  1993-01-01
8       KENYA   AFRICA  ECONOMY ANODIZED TIN
9       papaya
12      SHIP    MAIL    1994-01-01
14      1994-04-01
7       PERU    KENYA
4       1997-05-01
3       BUILDING        1995-03-28
20      almond  1996-01-01      GERMANY
16      Brand#34        STANDARD BURNISHED      38      20      46      7
        31      48      28      22
6       1993-01-01      0.07    25
22      18      13      33      11      32      12      21
10      1993-08-01
13      unusual packages
```

```
2       15      NICKEL  MIDDLE EAST
21      PERU
18      312
11      EGYPT   0.0000001000
```

Appendix E : Refresh Function Source Code

CreateRF1Proc.sql

```
--  File:     CREATERF1PROC.SQL
--            Microsoft TPC-H Benchmark Kit Ver. 1.00
--            Copyright Microsoft, 1999
--

IF exists (SELECT name FROM sysobjects WHERE name = 'RF1')
       DROP PROCEDURE RF1
GO

--
--  Create a stored RefreshInsert procedure which will catch the
deadlock
--  victim abort and restart the insert transaction.
--
CREATE PROCEDURE RF1
       @current_execution INTEGER, @insert_sets INTEGER,
@parallel_executions INTEGER, @total_executions INTEGER
AS
BEGIN

DECLARE @startdate DATETIME
DECLARE @enddate DATETIME
DECLARE @edate DATETIME
DECLARE @rangeStart INTEGER
DECLARE @rangeSize INTEGER
DECLARE @range INTEGER

DECLARE @success INTEGER
DECLARE @index INTEGER
DECLARE @div INTEGER
DECLARE @mod INTEGER
DECLARE @skip INTEGER
DECLARE @i INTEGER
DECLARE @rangeSum INTEGER
DECLARE @totRangeSize INTEGER
DECLARE @stmt NCHAR(1000)
DECLARE @orderSql NCHAR(1000)
DECLARE @liSql NCHAR(1000)

SET @skip = @total_executions/@parallel_executions
SET @div = (@current_execution - 1)/@parallel_executions
SET @mod = (@current_execution - 1) - @div * @parallel_executions
SET @index = @mod*@skip + @div + 1
--
-- Get the range for this execution
--
SET @stmt = N'SELECT @sdate = dateadd(day,-1,min(O_ORDERDATE)), @edate =
max(O_ORDERDATE)
              FROM NEWORDERS'
EXEC sp_executesql @stmt,N'@sdate datetime output, @edate datetime
output',@startdate output, @enddate output

IF (@total_executions > @parallel_executions)
       BEGIN
       SET @div = (@index-1)/@skip
       SET @mod = (@index-1) - @div * @skip
       SET @rangeSize = datediff(day, @startdate,
@enddate)/@parallel_executions + 1
       SET @totRangeSize = @rangeSize
       SET @rangeSum = 0
       SET @rangeStart = @div * @rangeSize
       SET @i = @mod
       while (@i > 0)
              BEGIN
              SET @rangeSize = (@totRangeSize - @rangeSum)/2
              SET @rangeSum = @rangeSum + @rangeSize
              SET @rangeStart = @rangeStart + @rangeSize
              SET @insert_sets = @insert_sets/2
              SET @i = @i - 1
              end

       IF (@mod + 1 = @skip) --  last allocation
              SET @rangeSize = @totRangeSize - @rangeSum
       ELSE
              SET @rangeSize = (@totRangeSize - @rangeSum)/2
       IF (@rangeSize < 0)
              SET @rangeSize = 0
       IF (@insert_sets <= 0)
              SET @insert_sets = 1
       end
ELSE
       BEGIN
       SET @rangeSize = datediff(day, @startdate,
@enddate)/@total_executions
       SET @rangeStart = @rangeSize * (@index - 1)
       end

SET @startdate = dateadd(day, @rangeStart, @startdate)
IF (@index < @total_executions)
       SET @enddate = dateadd(day, @rangeSize, @startdate)

SET @range = datediff(day, @startdate, @enddate) / @insert_sets

--
--  This handles the case when the max-min/insert_sets is less that 1
--
IF @range = 0
```

```
	SET @range = 1

--
-- Generate the two insert statements
--
SET @edate = dateadd(day, @range, @startdate)
SET @orderSql = N'INSERT INTO ORDERS (O_ORDERKEY, O_CUSTKEY,
O_ORDERSTATUS, O_TOTALPRICE,
                                      O_ORDERDATE, O_ORDERPRIORITY,
O_CLERK, O_SHIPPRIORITY, O_COMMENT)
                     (SELECT O_ORDERKEY, O_CUSTKEY, O_ORDERSTATUS,
O_TOTALPRICE,
                                      O_ORDERDATE, O_ORDERPRIORITY,
O_CLERK, O_SHIPPRIORITY, O_COMMENT
                      FROM NEWORDERS
                      WHERE O_ORDERDATE > @startdate AND O_ORDERDATE <=
@edate)
                     option (loop join)'
SET @liSql = N'INSERT INTO LINEITEM
(L_ORDERKEY,L_PARTKEY,L_SUPPKEY,L_LINENUMBER,L_QUANTITY,
                              L_EXTENDEDPRICE, L_DISCOUNT, L_TAX,
L_RETURNFLAG, L_LINESTATUS,
                              L_SHIPDATE, L_COMMITDATE, L_RECEIPTDATE,
L_SHIPINSTRUCT, L_SHIPMODE, L_COMMENT)
                     (SELECT
L_ORDERKEY,L_PARTKEY,L_SUPPKEY,L_LINENUMBER,L_QUANTITY,
                              L_EXTENDEDPRICE, L_DISCOUNT, L_TAX,
L_RETURNFLAG, L_LINESTATUS,
                              L_SHIPDATE, L_COMMITDATE, L_RECEIPTDATE,
L_SHIPINSTRUCT, L_SHIPMODE, L_COMMENT
                      FROM NEWLINEITEM, NEWORDERS
                      WHERE L_ORDERKEY = O_ORDERKEY AND O_ORDERDATE >
@startdate AND
                                      O_ORDERDATE <= @edate)
                     option (loop join)'

--
--  Loop through the order keys inserting sets into the
--  ORDERS and LINTEITEM tables
--
WHILE @startdate < @enddate
	BEGIN
	--
	--  Insert into ORDERS and LINEITEM tables
	--
	INSERT_TRANS:
	SET @success = 1
	BEGIN TRANSACTION

	BEGIN TRY
		EXEC sp_executesql @orderSql, N'@startdate datetime,
@edate datetime', @startdate, @edate
		EXEC sp_executesql @liSql, N'@startdate datetime, @edate
datetime', @startdate, @edate
	END TRY
	BEGIN CATCH
		SET @success = 0
		IF (error_number() = 1205)	-- deadlock victim
			PRINT 'Insert deadlock - restarting RF1'
		ELSE
			BEGIN			-- not a deadlock
			PRINT 'Error - Not a deadlock'
			PRINT ERROR_NUMBER()
			PRINT ERROR_SEVERITY()
			PRINT ERROR_MESSAGE()
			PRINT ERROR_STATE()
			PRINT XACT_STATE()
			END
		IF (@@trancount>0)
			ROLLBACK TRANSACTION
	END CATCH

	IF (@success = 0)			-- deadlock - redo the
inserts
		GOTO INSERT_TRANS

	COMMIT TRANSACTION

	SET @startdate = @edate
	SET @edate = dateadd(day, @range, @edate)

	IF (@edate > @enddate)
		SET @edate = @enddate

	END
END

GO
```

CreateRF2Proc.sql

```
--  File:	CREATERF2PROC.SQL
--		Microsoft TPC-H Benchmark Kit Ver. 1.00
--		Copyright Microsoft, 1999
--

IF exists (SELECT name FROM sysobjects WHERE name = 'RF2')
	DROP PROCEDURE RF2
GO

--
--  Create a stored Refresh Delete procedure which will catch the
deadlock
--  victim abort and restart the delete transaction.
--
CREATE PROCEDURE RF2
       @current_execution INTEGER, @delete_sets INTEGER,
@parallel_executions INTEGER, @total_executions INTEGER
AS
BEGIN

DECLARE @startdate DATETIME
```

```
DECLARE @enddate DATETIME
DECLARE @edate DATETIME
DECLARE @rangeStart INTEGER
DECLARE @rangeSize INTEGER
DECLARE @range INTEGER

declare @success INTEGER
declare @index INTEGER
declare @div INTEGER
declare @mod INTEGER
declare @skip INTEGER
declare @i INTEGER
declare @rangeSum INTEGER
declare @totRangeSize INTEGER
declare @sql NCHAR(1000)
declare @orderSql NCHAR(1000)
declare @liSql NCHAR(1000)

SET @skip = @total_executions/@parallel_executions
SET @div = floor((@current_execution-1)/@parallel_executions)
SET @mod = (@current_execution - 1) - @div * @parallel_executions
SET @index = @mod*@skip + @div + 1

SET @sql = N'SELECT @sdate = dateadd(day,-1,min(O_ORDERDATE)), @edate =
max(O_ORDERDATE)
              FROM MOD_OLDORDERS'
EXEC sp_executesql @sql,N'@sdate datetime output, @edate datetime
output',@startdate output, @enddate output

IF (@total_executions > @parallel_executions)
       BEGIN
       SET @div = (@index-1)/@skip
       SET @mod = (@index-1) - @div * @skip
       SET @rangeSize = datediff(day, @startdate,
@enddate)/@parallel_executions + 1
       SET @totRangeSize = @rangeSize
       SET @rangeSum = 0
       SET @rangeStart = @div * @rangeSize
       SET @i = @mod
       WHILE (@i > 0)
              BEGIN
              SET @rangeSize = (@totRangeSize - @rangeSum)/2
              SET @rangeSum = @rangeSum + @rangeSize
              SET @rangeStart = @rangeStart + @rangeSize
              SET @delete_sets = @delete_sets/2
              SET @i = @i - 1
              END

       IF (@mod + 1 = @skip) --  last allocation
              SET @rangeSize = @totRangeSize - @rangeSum
       ELSE
              SET @rangeSize = (@totRangeSize - @rangeSum)/2
       IF (@rangeSize < 0)
              SET @rangeSize = 0
       IF (@delete_sets <= 0)
              SET @delete_sets = 1
       END
ELSE
       BEGIN
       SET @rangeSize = datediff(day, @startdate,
@enddate)/@total_executions
       SET @rangeStart = @rangeSize * (@index - 1)
       END

SET @startdate = dateadd(day, @rangeStart, @startdate)
IF (@index < @total_executions)
       SET @enddate = dateadd(day, @rangeSize, @startdate)

SET @range = datediff(day, @startdate, @enddate) / @delete_sets

--
--  This handles the case when the max-min/delete_sets is less that 1
--
IF @range = 0
       SET @range = 1


--
--  Loop through the order keys deleting sets from orders
--  and lineitem tables
--
SET @edate = dateadd(day, @range, @startdate)
SET @liSql = N'DELETE FROM LINEITEM WHERE L_ORDERKEY in
                     (SELECT O_ORDERKEY FROM MOD_OLDORDERS
                      WHERE O_ORDERDATE > @startdate AND O_ORDERDATE <=
@edate)
                     option (loop join)'
SET @orderSql = N'DELETE FROM ORDERS WHERE O_ORDERKEY in
                     (SELECT O_ORDERKEY FROM MOD_OLDORDERS
                      WHERE O_ORDERDATE > @startdate AND O_ORDERDATE <=
@edate)
                     option (loop join)'

WHILE @startdate < @enddate
       BEGIN

       DELETE_TRANS:
       SET @success = 1
       BEGIN TRANSACTION

       BEGIN TRY
              EXEC sp_executesql @liSql, N'@startdate datetime, @edate
datetime', @startdate, @edate
              EXEC sp_executesql @orderSql, N'@startdate datetime,
@edate datetime', @startdate, @edate
       END TRY
       BEGIN CATCH
              SET @success = 0
              IF (error_number() = 1205)    -- deadlock victim
                     PRINT 'Insert deadlock - restarting RF2'
              ELSE
                     BEGIN                  -- not a deadlock
                     PRINT 'Error - Not a deadlock'
```

```
                        PRINT ERROR_NUMBER()
                        PRINT ERROR_SEVERITY()
                        PRINT ERROR_MESSAGE()
                        PRINT ERROR_STATE()
                        PRINT XACT_STATE()
                        END
                IF (@@trancount>0)
                        ROLLBACK TRANSACTION
        END CATCH

        IF (@success = 0)                       -- deadlock - redo the
inserts
                GOTO DELETE_TRANS

        COMMIT TRANSACTION

        SET @startdate = @edate
        SET @edate = dateadd(day, @range, @edate)

        IF (@edate > @enddate)
                SET @edate = @enddate

        END
END

GO
```

RF1_index.sql

```
create clustered index NEWORDERS_CLUIDX
        on NEWORDERS (O_ORDERDATE)
        on GENERAL_FG

create clustered index NEWLINEITEM_CLUIDX
        on NEWLINEITEM (L_ORDERKEY)
        on GENERAL_FG
```

RF1_init.sql

```
--
-- Create tables to hold RF1 input data
--

if exists (select name from sysobjects where name = 'NEWORDERS')
                        drop table NEWORDERS

create table NEWORDERS (O_ORDERKEY bigint not null,
                                O_CUSTKEY int not null,
                                O_ORDERSTATUS char(1)  not null,
                                O_TOTALPRICE  float  not null,
                                O_ORDERDATE datetime not null,
                                O_ORDERPRIORITY char(15)  not null,
                                O_CLERK  char(15) not null,
                                O_SHIPPRIORITY  int not null,
                                O_COMMENT varchar(79) not null) on
GENERAL_FG


if exists (select name from sysobjects  where name = 'NEWLINEITEM')
                        drop table NEWLINEITEM
create table NEWLINEITEM (L_ORDERKEY  bigint  not null,
                                L_PARTKEY int not null,
                                L_SUPPKEY int not null,
                                L_LINENUMBER int not null,
                                L_QUANTITY float not null,
                                L_EXTENDEDPRICE float not null,
                                L_DISCOUNT float not null,
                                L_TAX float not null,
                                L_RETURNFLAG char(1) not null,
                                L_LINESTATUS char(1) not null,
                                L_SHIPDATE datetime not null,
                                L_COMMITDATE datetime not null,
                                L_RECEIPTDATE datetime not null,
                                L_SHIPINSTRUCT char(25)  not null,
                                L_SHIPMODE char(10 ) not null,
                                L_COMMENT varchar(44) not null) on
GENERAL_FG
```

RF1_load.sql

```
--
-- Load the insert data for the given segment
--

declare @sql nchar(1000)
declare @set integer

select @set = updateset from TPCH_AUX_TABLE

set @sql = 'bulk insert NEWLINEITEM
                from ''%RF_FLATFILE_DIR%\Lineitem.tbl.u' +
RTRIM(CONVERT(varchar(30),@set))
                        + '.%SEGMENT%'' with (FieldTerminator = ''|'',
RowTerminator =''\n'',tablock)'
exec sp_executesql @sql

set @sql = 'bulk insert NEWORDERS
                from ''%RF_FLATFILE_DIR%\Orders.tbl.u' +
RTRIM(CONVERT(varchar(30), @set))
                        + '.%SEGMENT%'' with (FieldTerminator = ''|'',
RowTerminator =''\n'',tablock)'
exec sp_executesql @sql
```

RF2_index.sql

```
create clustered index MOD_OLDORDERS_CLUIDX on MOD_OLDORDERS
(O_ORDERDATE)

create index MOD_OLDORDERS_IDX on MOD_OLDORDERS (O_ORDERKEY)
```

RF2_init.sql

```
--
-- Create the tables to hold input data for RF2
--

declare @segment integer
declare @sql nchar(1000)


if exists (select name from sysobjects  where name = 'MOD_OLDORDERS')
                       drop table MOD_OLDORDERS
create table MOD_OLDORDERS (O_ORDERDATE datetime, O_ORDERKEY bigint) on
GENERAL_FG


set @segment = 1
while @segment <= %FILES_PER_UPDATE_SET%
       begin
       set @sql = 'if exists (select name from sysobjects  where name =
''OLDORDERS_'
                                      +
RTRIM(CONVERT(varchar(30),@segment))+ ''')
                       drop table OLDORDERS_' +
RTRIM(CONVERT(varchar(30),@segment))
       exec sp_executesql @sql
       set @sql = 'create table OLDORDERS_' +
RTRIM(CONVERT(varchar(30),@segment))
                       + '(O_ORDERKEY int) on GENERAL_FG'
       exec sp_executesql @sql

       set @segment = @segment + 1
       end
```

RF2_load.sql

```
--
-- Load the RF2 data for the given segment
--

declare @set integer
declare @sql nchar(1000)

select @set = updateset from TPCH_AUX_TABLE

set @sql = 'bulk insert OLDORDERS_%SEGMENT%
               from ''%RF_FLATFILE_DIR%\Delete.u' +
RTRIM(CONVERT(varchar(30),@set))
                       + '.%SEGMENT%'' with (RowTerminator
=''|'',tablock)'
exec sp_executesql @sql

insert into MOD_OLDORDERS (O_ORDERDATE, O_ORDERKEY)
               (select B.O_ORDERDATE, B.O_ORDERKEY
               from OLDORDERS_%SEGMENT% A, ORDERS B
               where A.O_ORDERKEY = B.O_ORDERKEY)
```

Appendix F : Implementation Specific Layer and Source Code

F.1 Source Code

F.1.1 ExecuteDll

Execute.cpp

```
//  FILE:       Execute.cpp
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Implementation of CExecute.
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
#include "stdafx.h"

#include "ExecuteDll.h"
#include "SMExecute.h"
#include "Execute.h"

extern SQLHENV henv;

extern SM_Connection_Info     *p_Connections;
       // Pointer to open connections
extern int                                    iConnectionCount;
               // Number of open connections
extern CRITICAL_SECTION               hConnections;
       // Critical section to serialize access to available connections

#ifdef _TPCH_AUDIT
       extern FILE *pfLogFile;
                                      // Log file containing timestamps
       extern CRITICAL_SECTION hLogFileWrite;
               // Handle to critical section
#endif

////////////////////////////////////////////////////////////////////////
/////
// CExecute

char * g_szOdbcOps[] = {
               "SQLAllocHandle",
               "SQLDriverConnect",
               "SQLExecDirect",
```

```
                "SQLSetStmtAttr",
                "SQLCancel",
                "SQLNumResultCols",
                "SQLDescribeCol",
                "SQLColAttribute",
                "SQLFetch",
                "SQLGetData",
                "SQLRowCount",
                "SQLMoreResults",
                "SQLBindCol"
        };

char * CExecError::m_szExecErrorDesc[] = {
                "Connection is already in use."
        };

STDMETHODIMP CExecute::InterfaceSupportsErrorInfo(REFIID riid)
{
        static const IID* arr[] =
        {
                &IID_IExecute
        };
        for (int i=0; i < sizeof(arr) / sizeof(arr[0]); i++)
        {
                if (InlineIsEqualGUID(*arr[i],riid))
                        return S_OK;
        }
        return S_FALSE;
}

STDMETHODIMP CExecute::put_OutputFile(BSTR newVal)
{
        assert(m_pOutputFile);
        m_OutputFile = newVal;

        HRESULT hr = m_pOutputFile->put_FileName(newVal);
        if FAILED(hr)
        {
                m_pOutputFile->Release();
                m_pOutputFile = NULL;
        }
        return hr;
}

//DEL STDMETHODIMP CExecute::put_LogFile(BSTR newVal)
//DEL {
//DEL   assert(m_pLogFile);
//DEL
//DEL   m_pLogFile->put_FileName(newVal);
//DEL   return S_OK;
//DEL }

STDMETHODIMP CExecute::put_ErrorFile(BSTR newVal)
{
        assert(m_pErrorFile);
        m_ErrorFile = newVal;

        HRESULT hr = m_pErrorFile->put_FileName(newVal);
        if FAILED(hr)
        {
                m_pErrorFile->Release();
                m_pErrorFile = NULL;
        }
        return hr;
}

STDMETHODIMP CExecute::DoExecute(BSTR szCommand, BSTR szExecutionDtls,
ExecutionType ExecMethod, \
                                                                BOOL
bNoCount, BOOL bNoExecute, BOOL bParseOnly, BOOL bQuotedIds, \
                                                                BOOL
bAnsiNulls, BOOL bShowQP, BOOL bStatsTime, BOOL bStatsIO, \
                                                                long
lRowCount, long lQueryTmout, BSTR szConnection)
{
    HANDLE              hThrd;
    DWORD               tid;

   _CrtSetReportFile(_CRT_WARN, _CRTDBG_FILE_STDOUT);

        m_szCommand = szCommand;
        m_szExecDtls = szExecutionDtls;

        m_ExecMthd = ExecMethod;
        if (m_ExecMthd == execODBC)
    {
         m_bNoCount = bNoCount;
                m_bNoExecute = bNoExecute;
                m_bParseOnly = bParseOnly;
                m_bQuotedIds = bQuotedIds;
                m_bAnsiNulls = bAnsiNulls;
                m_bShowQP = bShowQP;
                m_bStatsTime = bStatsTime;
                m_bStatsIO = bStatsIO;
                m_lRowCount = lRowCount;
                m_lQueryTmout = lQueryTmout;
                m_szConnection = szConnection;
    }

        if((hThrd = CreateThread( 0, 0,
(LPTHREAD_START_ROUTINE)ExecutionThread,
                        this, 0, &tid)) == NULL)
                return(RaiseSystemError());

        CloseHandle(hThrd);

        return S_OK;
}

STDMETHODIMP CExecute::Abort()
{
        if (m_ExecMthd == execShell)
```

```
                return(AbortShell());
        else
                return(AbortODBC());
}

void ExecutionThread(LPVOID lpParameter)
{

        CExecute        *MyExecute = (CExecute*)lpParameter;

        MyExecute->m_tElapsedTime = 0;

        GetLocalTime(&MyExecute->m_tStartTime);
        MyExecute->PostMessage(WM_TASK_START, 0, 0);

#ifdef _TPCH_AUDIT
        char            szBuffer[MAXLOGCMDBUF];
        char            szFmt[MAXBUFLEN];

       sprintf(szFmt, "Start Step: '%%.%ds' at '%d/%d/%d %d:%d:%d:%d'\n",
                MAXLOGCMDLEN,
                MyExecute->m_tStartTime.wMonth, MyExecute-
>m_tStartTime.wDay,
                MyExecute->m_tStartTime.wYear, MyExecute-
>m_tStartTime.wHour,
                MyExecute->m_tStartTime.wMinute, MyExecute-
>m_tStartTime.wSecond,
                MyExecute->m_tStartTime.wMilliseconds);
        if (MyExecute->m_ExecMthd == execShell)
                WriteFileToTpchLog((LPSTR)MyExecute->m_szCommand, szFmt);
        else
        {
                sprintf(szBuffer, szFmt, (LPSTR)MyExecute->m_szCommand);
                WriteToTpchLog(szBuffer);
        }
#endif

        // Initialize the run status for the step to running. The
completion status for
        // the step will be initialized by the Shell and ODBC execution
functions.
    MyExecute->m_StepStatus = gintRunning;

        if (MyExecute->m_ExecMthd == execShell)
                MyExecute->m_tElapsedTime = MyExecute->ExecuteShell();
        else
                MyExecute->m_tElapsedTime = MyExecute->ExecuteODBC();

        // Close the output, log and error files
        if (MyExecute->m_pOutputFile)
                MyExecute->m_pOutputFile->Release();
        MyExecute->m_pOutputFile = NULL;

        MyExecute->m_ExecTime = NULL;

        GetLocalTime(&MyExecute->m_tEndTime);

#ifdef _TPCH_AUDIT
        sprintf(szFmt, "Complete Step: '%%.%ds' at
'%d/%d/%d %d:%d:%d:%d'\n",
                MAXLOGCMDLEN,
                MyExecute->m_tEndTime.wMonth, MyExecute->m_tEndTime.wDay,
                MyExecute->m_tEndTime.wYear, MyExecute->m_tEndTime.wHour,
                MyExecute->m_tEndTime.wMinute, MyExecute-
>m_tEndTime.wSecond,
                MyExecute->m_tEndTime.wMilliseconds);
        if (MyExecute->m_ExecMthd == execShell)
                WriteFileToTpchLog((LPSTR)MyExecute->m_szCommand, szFmt);
        else
        {
                sprintf(szBuffer, szFmt, (LPSTR)MyExecute->m_szCommand);
                WriteToTpchLog(szBuffer);
        }
#endif

        MyExecute->PostMessage(WM_TASK_FINISH, 0, 0);

        return;
}

#ifdef _TPCH_AUDIT

void WriteFileToTpchLog(LPSTR szFile, LPSTR szFmt)
{
        // Reads a maximum of MAXLOGCMDBUF characters from the command
file and writes it to the log
        FILE    *fpCmd;
        int             iRead;
        char    szBuf[MAXLOGCMDBUF];
        char    szCmd[MAXLOGCMDLEN];

        if ( pfLogFile != NULL )
        {
                if ( (fpCmd = fopen(szFile, FILE_ACCESS_READ)) != NULL)
                {
                        iRead = fread(szCmd, sizeof(char), sizeof(szCmd) /
sizeof(char), fpCmd);
                        if (iRead < MAXLOGCMDLEN)
                                szCmd[iRead] = '\0';
                        else
                                szCmd[MAXLOGCMDLEN - 1] = '\0';
                        sprintf(szBuf, szFmt, szCmd);
                        WriteToTpchLog(szBuf);
                        fclose(fpCmd);
                }
        }
}

void WriteToTpchLog(char *szMsg)
{
        if (pfLogFile != NULL)
        {
```

```
                EnterCriticalSection(&hLogFileWrite);
                fprintf(pfLogFile, szMsg);
                LeaveCriticalSection(&hLogFileWrite);
        }

        return;
}
#endif

TC_TIME CExecute::ExecuteShell()
{
        STARTUPINFOA                    Start;
        PROCESS_INFORMATION             proc;
        DWORD                                   exitCode;
        TC_TIME                                 tElapsed = 0;
        _bstr_t                                 szCommand("cmd /c ");
        LPSTR                                   szStartDir;
        CURRENCY                                Elapsed;

        szCommand += m_szCommand;

        // Redirect output and error information
        szCommand += " > " + m_OutputFile + " 2> " + m_ErrorFile;

        // Initialize the STARTUPINFO structure:
        memset(&Start, 0, sizeof(STARTUPINFOA)) ;
        Start.cb                = sizeof(Start);
        Start.dwFlags           = STARTF_USESHOWWINDOW;
        Start.wShowWindow       = SW_SHOWMINNOACTIVE;

        memset(&proc, 0, sizeof(PROCESS_INFORMATION)) ;

        szStartDir = strcmp((LPCTSTR)m_szExecDtls, "") == 0 ? NULL :
(LPSTR)m_szExecDtls;

        m_ExecTime->Start();

        // Start the shelled application:
        if (!CreateProcessA( NULL, (LPSTR)szCommand, NULL, NULL, FALSE,
                NORMAL_PRIORITY_CLASS, NULL, szStartDir, &Start, &proc ))
        {
                m_StepStatus = gintFailed;
                LogSystemError(m_pErrorFile);

                m_ExecTime->Stop(&Elapsed);
                return((TC_TIME)Elapsed.int64);
        }

        m_hHandle = proc.hProcess;
        // Give the process time to execute and finish
        WaitForSingleObject(m_hHandle, INFINITE);
        m_ExecTime->Stop(&Elapsed);

        if (!GetExitCodeProcess(m_hHandle, &exitCode))
        {
                m_StepStatus = gintFailed;
                LogSystemError(m_pErrorFile);
        }
        else
                m_StepStatus = gintComplete;

        // Close all open handles to the shelled process
        CloseHandle(m_hHandle);

        return((TC_TIME)Elapsed.int64);
}

STDMETHODIMP CExecute::AbortShell()
{
        if (m_hHandle != SQL_NULL_HSTMT)
                if (!TerminateProcess(m_hHandle, 0))
                        return(RaiseSystemError());

        return(S_OK);
}

TC_TIME CExecute::ExecuteODBC()
{
        TC_TIME                         tElapsed = 0;
        HDBC                            m_hdbc;
        SQLRETURN                       rc;
        LPSTR                           szCmd;
        CURRENCY                        Elapsed;
        BOOL                            bDoConnect = FALSE;

        // ODBC specific initialization
        m_hdbc = SQL_NULL_HDBC;

        try
        {
                // Allocate a new connection if we are creating a dynamic
connection or if
                // the named connection doesn't exist
                InitializeConnection(&m_hdbc, &bDoConnect);

                // Ensure that the connection is valid.
#ifdef _DEBUG
                _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLDriverConnect.\n");
#endif

                if (bDoConnect)
                {
                        // Allocate connection handle, open a connection
and set connection attributes.
#ifdef _DEBUG
                        _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLDriverConnect.\n");
#endif

                        if (m_bAbort)
                                return(tElapsed);
```

```
                        // Connect to the server using the passed in
connection string
                        rc = SQLDriverConnect(m_hdbc, NULL,
                                (unsigned char *)(LPSTR)m_szExecDtls,
SQL_NTS,
                                NULL, 0, NULL, SQL_DRIVER_NOPROMPT);
                        HandleODBCError(rc, SQL_HANDLE_DBC, m_hdbc,
SMSQLDriverConnect);
                }

                ReConnectDeadConnection(&m_hdbc);

#ifdef _DEBUG
                _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLAllocHandle for hdbc.\n");
#endif

                if (!m_bAbort && (rc = SQLAllocHandle(SQL_HANDLE_STMT,
m_hdbc, &m_hHandle)) != SQL_SUCCESS)
                        HandleODBCError(rc, SQL_HANDLE_DBC, m_hdbc,
SMSQLAllocHandle);

                // Set connection attributes if any have been modified
from the default values
                if (m_lRowCount > 0)
                {
                        char                    szConnOptions[512];

                        sprintf(szConnOptions, "SET ROWCOUNT %d ",
m_lRowCount);
                        SetConnectionOption(szConnOptions, &m_hdbc);
                }

                if (m_bQuotedIds)
                        SetConnectionOption("SET QUOTED_IDENTIFIER ON ",
&m_hdbc);

                if (!m_bAnsiNulls)
                        SetConnectionOption("SET ANSI_NULL_DFLT_OFF ON ",
&m_hdbc);

                if (!m_bAbort && m_lQueryTmout > 0)
                {
#ifdef _DEBUG
                        _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLSetStmtAttr.\n");
#endif

                        // Set the query timeout on the statement handle
                        rc = SQLSetStmtAttr(m_hHandle,
SQL_ATTR_QUERY_TIMEOUT, &m_lQueryTmout,
                                SQL_IS_UINTEGER);
                        HandleODBCError(rc, SQL_HANDLE_STMT, m_hHandle,
SMSQLSetStmtAttr);
                }

                if (m_bNoExecute)
                        SetConnectionOption("SET NOEXEC ON ", &m_hdbc);
                else if (m_bParseOnly)
                        SetConnectionOption("SET PARSEONLY ON ", &m_hdbc);
                else if (m_bShowQP)
                        // Important to ensure that this is the last
connection attributes being set -
                        // otherwise showplans are generated for all
remaining SET statements
                        SetConnectionOption("SET SHOWPLAN_TEXT ON ",
&m_hdbc);
                else
                {
                        if (m_bNoCount)
                                SetConnectionOption("SET NOCOUNT ON ",
&m_hdbc);

                        if (m_bStatsIO)
                                SetConnectionOption("SET STATISTICS IO ON
", &m_hdbc);

                        // Important to ensure that this is the last
connection attributes being set -
                        // otherwise timing statistics are generated for
all remaining SET statements
                        if (m_bStatsTime)
                                SetConnectionOption("SET STATISTICS TIME
ON ", &m_hdbc);
                }

                m_szCmd = (LPSTR)m_szCommand;
                m_ExecTime->Start();

                while ((szCmd = NextCmdInBatch((LPSTR)m_szCommand)) !=
NULL && !m_bAbort)
                {
#ifdef _DEBUG
                        _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLExecDirect.\n");
#endif

                        // Execute the ODBC command
                        rc = SQLExecDirect(m_hHandle, (unsigned char
*)szCmd, SQL_NTS);
                        HandleODBCError(rc, SQL_HANDLE_STMT, m_hHandle,
SMSQLExecDirect);

                        free(szCmd);

                        // Call a procedure to log the results to the
output file
                        ProcessResultsets();

                }
                m_ExecTime->Stop(&Elapsed);
```

```
                ResetConnectionProperties(&m_hdbc);
        }
        catch(CODBCError *pErr)
        {
                m_StepStatus = gintFailed;
                delete pErr;
        }
        catch(CExecError *pErr)
        {
                m_StepStatus = gintFailed;
                pErr->LogErrors(this);
                delete pErr;
        }

        ODBCCleanup(&m_hdbc, &m_hHandle);

        if (m_StepStatus != gintFailed)
                m_StepStatus = gintComplete;

        return((DWORD)Elapsed.int64);
}

void CExecute::InitializeConnection(HDBC *phdbc, BOOL *pbDoConnect)
{
        SQLRETURN                       rc;

        *pbDoConnect = TRUE;

        if (IsDynamicConnection())
        {
#ifdef _DEBUG
                _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLAllocHandle for m_hdbc.\n");
#endif

                rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, phdbc);
                HandleODBCError(rc, SQL_HANDLE_ENV, henv,
SMSQLAllocHandle);

                return;
        }

        EnterCriticalSection(&hConnections);
        // Returns the connection handle if the connection,
m_szConnection, exists
        for (m_iConnectionIndex = iConnectionCount - 1;
m_iConnectionIndex >= 0; m_iConnectionIndex--)
        {
                if (!strcmp( (p_Connections + m_iConnectionIndex)-
>szConnectionName, (LPSTR)m_szConnection))
                {
                        if (!(p_Connections + m_iConnectionIndex)->bInUse)
                        {
                                *phdbc = (p_Connections +
m_iConnectionIndex)->hdbc;
                                (p_Connections + m_iConnectionIndex)-
>bInUse = TRUE;

                                *pbDoConnect = FALSE;
                                break;
                        }
                        else
                        {
                                LeaveCriticalSection(&hConnections);

                                throw new
CExecError(CExecError::SM_ERR_CONN_IN_USE);
                        }
                }
        }

        if (m_iConnectionIndex < 0)
        {
                // Connection was not found. Allocate connection handle
and add it to list of
                // available connections.
#ifdef _DEBUG
                _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLAllocHandle for m_hdbc.\n");
#endif

                rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, phdbc);
                HandleODBCError(rc, SQL_HANDLE_ENV, henv,
SMSQLAllocHandle);

                m_iConnectionIndex = iConnectionCount++;

                p_Connections = (SM_Connection_Info
*)realloc(p_Connections, iConnectionCount * sizeof(SM_Connection_Info));

                strcpy((p_Connections + m_iConnectionIndex)-
>szConnectionName, (LPSTR)m_szConnection);
                (p_Connections + m_iConnectionIndex)->hdbc = *phdbc;
                (p_Connections + m_iConnectionIndex)->bInUse = TRUE;
        }

        LeaveCriticalSection(&hConnections);

        return;
}

void CExecute::ReConnectDeadConnection(HDBC *phdbc)
{
        SQLRETURN               rc;
        SQLUINTEGER             uConnDead;

        // Connect to the server using the passed in connection string
        rc = SQLGetConnectAttr(*phdbc, SQL_ATTR_CONNECTION_DEAD,
                &uConnDead, SQL_IS_UINTEGER, NULL);
        HandleODBCError(rc, SQL_HANDLE_DBC, *phdbc, SMSQLDriverConnect);
```

```
	if (uConnDead == SQL_CD_TRUE)
	{
		// Cleanup the old connection and re-connect.
#ifdef _DEBUG
		_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLDisconnect.\n");
#endif
		rc = SQLDisconnect(*phdbc);
#ifdef _DEBUG
		_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLFreeHandle for hdbc.\n");
#endif
		SQLFreeHandle(SQL_HANDLE_DBC, *phdbc);
		*phdbc = SQL_NULL_HDBC;

#ifdef _DEBUG
		_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLDriverConnect.\n");
#endif

		rc = SQLAllocHandle(SQL_HANDLE_DBC, henv, phdbc);
		HandleODBCError(rc, SQL_HANDLE_ENV, henv,
SMSQLAllocHandle);

		// Connect to the server using the passed in connection
string
		rc = SQLDriverConnect(*phdbc, NULL,
			(unsigned char *)(LPSTR)m_szExecDtls, SQL_NTS,
			NULL, 0, NULL, SQL_DRIVER_NOPROMPT);
		HandleODBCError(rc, SQL_HANDLE_DBC, *phdbc,
SMSQLDriverConnect);
	}

	return;
}

void CExecute::ResetConnectionUsage()
{
	if(m_iConnectionIndex >= 0 && m_iConnectionIndex <
iConnectionCount)
	{
		EnterCriticalSection(&hConnections);
		(p_Connections + m_iConnectionIndex)->bInUse = FALSE;
		LeaveCriticalSection(&hConnections);
	}

	return;
}

void CExecute::ResetConnectionProperties(HDBC *p_hdbc)
{
	SQLRETURN			rc;

	// Reset connection attributes if any have been modified from the
default values
	if (m_bNoExecute)
		SetConnectionOption("SET NOEXEC OFF ", p_hdbc);
	else if (m_bParseOnly)
		SetConnectionOption("SET PARSEONLY OFF ", p_hdbc);
	else if (m_bShowQP)
		// Reset connection attributes in reverse order
		SetConnectionOption("SET SHOWPLAN_TEXT OFF ", p_hdbc);
	else
	{
		// Reset connection attributes in reverse order
		if (m_bStatsTime)
			SetConnectionOption("SET STATISTICS TIME OFF ",
p_hdbc);

		if (m_bNoCount)
			SetConnectionOption("SET NOCOUNT OFF ", p_hdbc);

		if (m_bStatsIO)
			SetConnectionOption("SET STATISTICS IO OFF ",
p_hdbc);
	}

	if (m_lRowCount > 0)
	{
		char			szConnOptions[512];

		sprintf(szConnOptions, "SET ROWCOUNT 0 ");
		SetConnectionOption(szConnOptions, p_hdbc);
	}

	if (m_bQuotedIds)
		SetConnectionOption("SET QUOTED_IDENTIFIER OFF ",
p_hdbc);

	if (!m_bAnsiNulls)
		SetConnectionOption("SET ANSI_NULL_DFLT_OFF OFF ",
p_hdbc);

	if (m_lQueryTmout > 0)
	{
		SQLUINTEGER		lQueryTmout = 0;

#ifdef _DEBUG
		_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLSetStmtAttr.\n");
#endif

		// Set the query timeout on the statement handle
		rc = SQLSetStmtAttr(m_hHandle, SQL_ATTR_QUERY_TIMEOUT,
&lQueryTmout,
			SQL_IS_UINTEGER);
		HandleODBCError(rc, SQL_HANDLE_STMT, m_hHandle,
SMSQLSetStmtAttr);
	}

	return;
```

```
}

LPSTR CExecute::NextCmdInBatch(LPSTR szBatch)
{
	LPSTR	szCmd, szSeparator, szStart;
	char	szNext;

	szStart = m_szCmd;

	while ( (szSeparator = strstr(szStart, CMD_SEPARATOR)) != NULL)
	{
		szNext = *(szSeparator + strlen(CMD_SEPARATOR));
		if ( szNext == '\n' || szNext == '\r' || szNext == '\0')
			break;
		else
			szStart = szSeparator + strlen(CMD_SEPARATOR);
	}

	if (!szSeparator)
	{
		// No more GO's
		if (strlen(m_szCmd) > 0)
		{
			szCmd = (LPSTR)malloc(strlen(m_szCmd) + 1);
			strcpy(szCmd, m_szCmd);
			m_szCmd += strlen(m_szCmd);
		}
		else
			szCmd = NULL;
	}
	else if (szSeparator - m_szCmd > 0)
	{
		// Strip the succeeding newline
		szCmd = (LPSTR)malloc(szSeparator - m_szCmd);
		strncpy(szCmd, m_szCmd, szSeparator - m_szCmd - 1);
		*(szCmd + (szSeparator - m_szCmd - 1)) = '\0';
		m_szCmd += szSeparator - m_szCmd + strlen(CMD_SEPARATOR);
		if ( szNext == '\n' || szNext == '\r')
			m_szCmd += 1;
	}
	else
		szCmd = NULL;

	return(szCmd);
}

void CExecute::SetConnectionOption(LPSTR szConn, HDBC *pHdbc)
{
	// Executes the passed in connection options 'set' statement.
Returns True if it succeeded
	char			szConnOptions[512];
	SQLRETURN		rc;

	sprintf(szConnOptions, szConn);

#ifdef _DEBUG
	_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing SQLExecDirect
for connection option.\n");
#endif

	if (m_bAbort)
		return;

	rc = SQLExecDirect(m_hHandle, (unsigned char *)szConnOptions,
SQL_NTS);
	HandleODBCError(rc, SQL_HANDLE_STMT, m_hHandle, SMSQLExecDirect);

	return;
}

STDMETHODIMP CExecute::AbortODBC()
{
	m_bAbort = TRUE;

	try
	{
		if (m_hHandle != SQL_NULL_HSTMT)
		{
#ifdef _DEBUG
			_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLCancel.\n");
#endif

			SQLRETURN rc = SQLCancel(m_hHandle);
			HandleODBCError(rc, SQL_HANDLE_STMT, m_hHandle,
SMSQLCancel);
		}
	}
	catch(CODBCError *pErr)
	{
		delete pErr;
	}

	return(S_OK);
}

void CExecute::ProcessResultsets()
{
    SQLSMALLINT		*CTypeArray, *CScaleArray;
    SQLINTEGER		*ColLenArray, *DispLenArray, *OffsetArray;
	SQLSMALLINT		iColNameLen, SQLType, iColNull, i, NumCols
= 0;
	SQLINTEGER		iDispLen, iRowCount;
    SQLRETURN		rc;
	char			szColName[MAX_DATA_LEN + 1];
    void			*DataPtr;
	SQLINTEGER		iLenOrInd = ALIGNBUF(sizeof(SQLINTEGER));
	SQLUINTEGER		iRowArraySize, iArrayElementSize;
//	SQLUINTEGER		NumRowsFetched;
//	SQLUSMALLINT	*RowStatusArray;
```

```
	if (!m_pOutputFile || m_bAbort)
		return;

	do
	{
#ifdef _DEBUG
		_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLNumResultCols.\n");
#endif

		// Determine the number of result set columns.
		rc = SQLNumResultCols(m_hHandle, &NumCols);
		HandleODBCError(rc, SQL_HANDLE_STMT, m_hHandle,
SMSQLNumResultCols);

		if (NumCols > 0)
		{

			// Allocate arrays to hold the C type, scale,
column and display length of the data
			CTypeArray = (SQLSMALLINT *) malloc(NumCols *
sizeof(SQLSMALLINT));
			CScaleArray = (SQLSMALLINT *) malloc(NumCols *
sizeof(SQLSMALLINT));
			ColLenArray = (SQLINTEGER *) malloc(NumCols *
sizeof(SQLINTEGER));
			DispLenArray = (SQLINTEGER *) malloc(NumCols *
sizeof(SQLINTEGER));

			OffsetArray = (SQLINTEGER *) malloc(NumCols *
sizeof(SQLINTEGER));
			OffsetArray[0] = 0;

			for (i = 0; i < NumCols && !m_bAbort; i++)
			{
#ifdef _DEBUG
				_CrtDbgReport(_CRT_WARN, NULL, 0, NULL,
"Executing SQLDescribeCol.\n");
#endif

				// Get the column description, include the
SQL type
				// Determine the column's byte length.
Calculate the offset in the buffer to the
				// data as the offset to the previous
column, plus the byte length of the previous
				// column, plus the byte length of the
previous column's length/indicator buffer.
				// Note that the byte length of the column
and the length/indicator buffer are increased
				// so that, assuming they start on an
alignment boundary, they will end on the byte
				// before the next alignment boundary.
Although this might leave some holes in the
				// buffer, it is a relatively inexpensive
way to guarantee alignment.
				rc = SQLDescribeCol(m_hHandle,
((SQLUSMALLINT) i)+1,
					(unsigned char *)szColName,
sizeof(szColName), &iColNameLen,
					&SQLType, (unsigned long
*)&ColLenArray[i], &CScaleArray[i], &iColNull);
				HandleODBCError(rc, SQL_HANDLE_STMT,
m_hHandle, SMSQLDescribeCol);

#ifdef _DEBUG
				_CrtDbgReport(_CRT_WARN, NULL, 0, NULL,
"Executing SQLColAttribute.\n");
#endif

				if (m_bAbort)
					return;

				rc = SQLColAttribute(m_hHandle,
((SQLUSMALLINT) i)+1, SQL_DESC_DISPLAY_SIZE, NULL, 0, NULL, &iDispLen);
				HandleODBCError(rc, SQL_HANDLE_STMT,
m_hHandle, SMSQLColAttribute);

				// GetDefaultCType contains a switch
statement that returns the default C type
				// for each SQL type.
				CTypeArray[i] = GetDefaultCType(SQLType);
				if ( (CTypeArray[i] == SQL_C_CHAR ||
CTypeArray[i] == SQL_C_BINARY) && ColLenArray[i] > MAX_DATA_LEN)
				{
					ColLenArray[i] = MAX_DATA_LEN;
					iDispLen = MAX_DATA_LEN;
				}

				DispLenArray[i] = max(iColNameLen,
iDispLen);
				DispLenArray[i] = max(DispLenArray[i],
sizeof(S_NULL));

				// Print the column names in the header
				PrintData(szColName, SQL_C_CHAR,
DispLenArray[i], 0, m_pOutputFile);

				// Add a byte for the null-termination
character
				ColLenArray[i] += 1;
				ColLenArray[i] = ALIGNBUF(ColLenArray[i]);

				// Calculate the offset in the buffer to
the data as the offset to the previous column,
				// plus the byte length of the previous
column, plus the byte length of the previous
				// column's length/indicator buffer.
				if (i)
					OffsetArray[i] = OffsetArray[i-1] +
ColLenArray[i-1] + iLenOrInd;
			}
```

```
                        m_pOutputFile->WriteLine(NULL);

                        iArrayElementSize = OffsetArray[NumCols-1] +
ColLenArray[NumCols-1] + iLenOrInd;
                        iRowArraySize = 1;
                        // Allocate the data buffer. The size of the
buffer is equal to the offset to the data
                        // buffer for the final column, plus the byte
length of the data buffer and length/indicator
                        // buffer for the last column.
                        DataPtr = malloc(iRowArraySize *
iArrayElementSize);

                        // Specify the size of the structure with the
SQL_ATTR_ROW_BIND_TYPE
                        // statement attribute. This also declares that
row-wise binding will
                        // be used. Declare the rowset size with the
SQL_ATTR_ROW_ARRAY_SIZE
                        // statement attribute. Set the
SQL_ATTR_ROW_STATUS_PTR statement
                        // attribute to point to the row status array. Set
the
                        // SQL_ATTR_ROWS_FETCHED_PTR statement attribute
to point to
                        // NumRowsFetched.
                        /*
                        RowStatusArray = (SQLUSMALLINT
*)malloc(iRowArraySize * sizeof(SQLUSMALLINT));

                        SQLSetStmtAttr(m_hHandle, SQL_ATTR_ROW_BIND_TYPE,
&iArrayElementSize, SQL_IS_UINTEGER);
                        SQLSetStmtAttr(m_hHandle, SQL_ATTR_ROW_ARRAY_SIZE,
&iRowArraySize, SQL_IS_UINTEGER);
                        SQLSetStmtAttr(m_hHandle, SQL_ATTR_ROW_STATUS_PTR,
RowStatusArray, SQL_IS_POINTER);
                        SQLSetStmtAttr(m_hHandle,
SQL_ATTR_ROWS_FETCHED_PTR, &NumRowsFetched, SQL_IS_POINTER);
                        */

                        // For each column, bind the address in the buffer
at the start of the memory allocated
                        // for that column's data and the address at the
start of the memory allocated for that
                        // column's length/indicator buffer.
                        for (i = 0; i < NumCols; i++)
                        {
                              SQLBindCol(m_hHandle, i + 1, CTypeArray[i],
(SQLPOINTER)((SQLCHAR *)DataPtr + OffsetArray[i]),
                                    ColLenArray[i], (SQLINTEGER
*)((SQLCHAR *)DataPtr + OffsetArray[i] + ColLenArray[i]));
                              HandleODBCError(rc, SQL_HANDLE_STMT,
m_hHandle, SMSQLBindCol);

                              // Underline each column name
                              memset(szColName, '-', DispLenArray[i]);
                              *(szColName + DispLenArray[i]) = '\0';
                              PrintData(szColName, SQL_C_CHAR,
DispLenArray[i], 0, m_pOutputFile);
                        }
                        m_pOutputFile->WriteLine(NULL);

#ifdef _DEBUG
                        _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLFetch.\n");
#endif

                        while (!m_bAbort && (rc = SQLFetch(m_hHandle)) !=
SQL_NO_DATA)
                        {
                              HandleODBCError(rc, SQL_HANDLE_STMT,
m_hHandle, SMSQLFetch);

                              /*
                              for (i = 0; i < NumRowsFetched; i++)
                              {
                                    if (RowStatusArray[i] ==
SQL_ROW_SUCCESS|| RowStatusArray[i] == SQL_ROW_SUCCESS_WITH_INFO)
                                    {
                                    */
                                          for (i = 0; i < NumCols;
i++)
                                          {
                                                // Retrieve and
print each row. PrintData accepts a pointer to the data, its C type,
                                                // and its byte
length/indicator.
                                                if ( *((SQLINTEGER
*)((SQLCHAR *)DataPtr + OffsetArray[i] + ColLenArray[i])) ==
SQL_NULL_DATA)
      PrintData(S_NULL, SQL_C_CHAR, DispLenArray[i], 0, m_pOutputFile);
                                                else

      PrintData((LPVOID)((SQLCHAR *)DataPtr + OffsetArray[i]),
CTypeArray[i],

      DispLenArray[i], CScaleArray[i], m_pOutputFile);
                                          }
                                          m_pOutputFile-
>WriteLine(NULL);
                                          /*
                                    }
                              }
                              */
                        }
                        m_pOutputFile->WriteLine(NULL);

                        /*
                        free(RowStatusArray);
                        */
```

```
                free(DataPtr);

                free(CTypeArray);
                free(CScaleArray);
                free(ColLenArray);
                free(DispLenArray);
        }

        // Write io statistics, if applicable
        LogODBCErrors(rc, SQL_HANDLE_STMT, m_hHandle,
SMSQLFetch);

#ifdef _DEBUG
        _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLRowCount.\n");
#endif

        if (m_bAbort)
                break;

        // action (insert, update, delete) query
        rc = SQLRowCount(m_hHandle, &iRowCount);
        HandleODBCError(rc, SQL_HANDLE_STMT, m_hHandle,
SMSQLRowCount);

        if (!m_bNoCount && iRowCount != -1)
        {
                sprintf(szColName, "(%d row(s) affected)",
iRowCount);
                _bstr_t temp(szColName);
                m_pOutputFile->WriteLine((BSTR)temp);
                m_pOutputFile->WriteLine(NULL);
        }

#ifdef _DEBUG
        _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLFreeStmt.\n");
#endif

        if (m_bAbort)
                break;

        SQLFreeStmt(m_hHandle, SQL_UNBIND);

#ifdef _DEBUG
        _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLMoreResults.\n");
#endif

        if (m_bAbort)
                break;

        // Process the next resultset. This function returns
'success with info' even
        // if there is no other resultset and there are
statistics messages to be printed.
        // Hence the check for -1 rows before printing.
        rc=SQLMoreResults(m_hHandle);
        if (rc != SQL_NO_DATA)
                HandleODBCError(rc, SQL_HANDLE_STMT, m_hHandle,
SMSQLMoreResults);

    } while (rc != SQL_NO_DATA);

        return;
}

void CExecute::PrintData(void *vData, SQLSMALLINT CType, SQLINTEGER
IndPtr, SQLSMALLINT iScale, ISMLog *pOutput)
{
        // PrintData accepts a pointer to the data, its C type,
        // and its byte length/indicator. It contains a switch statement
that casts and prints
        // the data according to its type.

        char            *s;
        char            fmt[MAXBUFLEN];
        int                     j = 0;
        SQLINTEGER      iColLen = IndPtr + 1;

        assert(iColLen);
        s = (LPSTR)malloc(iColLen + 1);

        if (s)
        {
                if (vData)
                {
                        switch(CType)
                        {
                        case SQL_C_CHAR:
                        case SQL_C_WCHAR:
                        case SQL_C_TYPE_DATE:
                        case SQL_C_TYPE_TIME:
                        case SQL_C_TYPE_TIMESTAMP:
                        case SQL_C_INTERVAL_YEAR:
                        case SQL_C_INTERVAL_MONTH:
                        case SQL_C_INTERVAL_YEAR_TO_MONTH:
                        case SQL_C_INTERVAL_DAY:
                        case SQL_C_INTERVAL_HOUR:
                        case SQL_C_INTERVAL_MINUTE:
                        case SQL_C_INTERVAL_SECOND:
                        case SQL_C_INTERVAL_DAY_TO_HOUR:
                        case SQL_C_INTERVAL_DAY_TO_MINUTE:
                        case SQL_C_INTERVAL_DAY_TO_SECOND:
                        case SQL_C_INTERVAL_HOUR_TO_MINUTE:
                        case SQL_C_INTERVAL_HOUR_TO_SECOND:
                        case SQL_C_INTERVAL_MINUTE_TO_SECOND:
                        case SQL_C_BINARY:
                                sprintf(fmt, "%%.%ds", iColLen);
                                j = sprintf(s, fmt, (char *)vData);
                                break;
```

```
                        case SQL_C_SHORT:
                                j = sprintf(s, "%d", *(short *)vData);
                                break;

                        case SQL_C_LONG:
                                j = sprintf(s, "%ld", *(long *)vData);
                                break;

                        case SQL_C_UBIGINT:
                                j = sprintf(s, "%I64d", *(__int64
*)vData);
                                break;

                        case SQL_C_FLOAT:
//                              sprintf(fmt, "%%.0%df", iScale);
                                j = sprintf(s, "%f", *(float *)vData);
                                break;

                        case SQL_C_DOUBLE:
//                              sprintf(fmt, "%%.0%df", iScale);
                                j = sprintf(s, "%f", *(double *)vData);
                                break;

                        case SQL_C_NUMERIC:
                                sprintf(fmt, "%%.0%df", iScale);
                                j = sprintf(s, fmt, *(double *)vData);
                                break;

                        default:
                                j = sprintf(s, "%s", vData);
                                break;
                        }
                }

                // Strip off terminating null character and pad the
string with blanks
                if (iColLen - j > 0)
                        memset(s + j, ' ', iColLen - j);

                *(s + iColLen) = '\0';

                // Write the field to the output file
                _bstr_t temp(s);
                pOutput->WriteField((BSTR)temp);
                free(s);
        }

        return;
}

SQLSMALLINT CExecute::GetDefaultCType(SQLINTEGER SQLType)
{
        // GetDefaultCType returns the C type for the passed in SQL
datatype.

    switch(SQLType)
    {
    case SQL_CHAR:
        case SQL_VARCHAR:
        case SQL_LONGVARCHAR:
        case SQL_WCHAR:
        case SQL_WVARCHAR:
        case SQL_WLONGVARCHAR:
             return(SQL_C_CHAR);

    case SQL_TINYINT:
                        return(SQL_C_CHAR);

    case SQL_SMALLINT:
                        return(SQL_C_SHORT);

    case SQL_INTEGER:
                        return(SQL_C_LONG);

    case SQL_BIGINT:
                        return(SQL_C_UBIGINT);

    case SQL_REAL:
                        return(SQL_C_FLOAT);

    case SQL_FLOAT:
    case SQL_DOUBLE:
//      case SQL_DECIMAL:
                        return(SQL_C_DOUBLE);

        case SQL_DECIMAL:
                        return(SQL_C_CHAR);

        case SQL_BIT:
                        return(SQL_C_CHAR);

    case SQL_BINARY:
        case SQL_VARBINARY:
        case SQL_LONGVARBINARY:
                        return(SQL_C_CHAR);
//                      return(SQL_C_BINARY);

    case SQL_TYPE_DATE:
                        return(SQL_C_CHAR);
//                      return(SQL_C_TYPE_DATE);

    case SQL_TYPE_TIME:
                        return(SQL_C_CHAR);
//                      return(SQL_C_TYPE_TIME);

    case SQL_TYPE_TIMESTAMP:
                        return(SQL_C_CHAR);
//                      return(SQL_C_TYPE_TIMESTAMP);

        case SQL_NUMERIC:
             return(SQL_C_FLOAT);
```

```
    case SQL_INTERVAL_YEAR:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_YEAR);

    case SQL_INTERVAL_MONTH:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_MONTH);

    case SQL_INTERVAL_YEAR_TO_MONTH:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_YEAR_TO_MONTH);

    case SQL_INTERVAL_DAY:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_DAY);

    case SQL_INTERVAL_HOUR:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_HOUR);

    case SQL_INTERVAL_MINUTE:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_MINUTE);

    case SQL_INTERVAL_SECOND:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_SECOND);

    case SQL_INTERVAL_DAY_TO_HOUR:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_DAY_TO_HOUR);

    case SQL_INTERVAL_DAY_TO_MINUTE:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_DAY_TO_MINUTE);

    case SQL_INTERVAL_DAY_TO_SECOND:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_DAY_TO_SECOND);

    case SQL_INTERVAL_HOUR_TO_MINUTE:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_HOUR_TO_MINUTE);

    case SQL_INTERVAL_HOUR_TO_SECOND:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_HOUR_TO_SECOND);

    case SQL_INTERVAL_MINUTE_TO_SECOND:
                    return(SQL_C_CHAR);
//                  return(SQL_C_INTERVAL_MINUTE_TO_SECOND);

       default:
              assert(TRUE);
              return(SQL_C_CHAR);
              break;
    }
}

/*
    FUNCTION: LogODBCErrors(SQLRETURN rc, SWORD fHandleType, SQLHANDLE
handle)
    COMMENTS: Formats ODBC errors or warnings and logs them. Also
initializes the
                        completion status for the step to failure, if an
ODBC error has occurred.
*/

void CExecute::LogODBCErrors(SQLRETURN nResult, SWORD fHandleType,
SQLHANDLE handle, OdbcOperations FailedOp)
{
       // Messages returned by the server (e.g. Print statements) will
be logged to the output file
       // ODBC warnings will be logged to the log file
       // All other ODBC errors will be logged to the error file.

       UCHAR          szErrState[SQL_SQLSTATE_SIZE+1];             //
SQL Error State string
       UCHAR          szErrText[SQL_MAX_MESSAGE_LENGTH+1];    // SQL
Error Text string
       char
       szBuffer[SQL_SQLSTATE_SIZE+SQL_MAX_MESSAGE_LENGTH+MAXBUFLEN+1] =
"";

                                       // formatted Error text Buffer
       SWORD          wErrMsgLen;
                      // Error message length
       SQLINTEGER     dwErrCode;
                      // Native Error code
       SQLRETURN      nErrResult;
                      // Return Code from SQLGetDiagRec
       SWORD          sMsgNum = 1;
              // Error sequence number
       _bstr_t        temp;

       if (IsErrorReturn(nResult))
       {
              sprintf(szBuffer, "ODBC Operation: '%s' returned error
code: %d",
                     g_szOdbcOps[FailedOp], nResult);
              temp = szBuffer;
              m_pErrorFile->WriteLine((BSTR) temp);
              m_StepStatus = gintFailed;
       }

       if (handle == SQL_NULL_HSTMT)
              return;

#ifdef _DEBUG
       _CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLGetDiagRec.\n");
#endif
```

```
	// call SQLGetDiagRec function with proper ODBC handles,
repeatedly until
	// function returns SQL_NO_DATA.
	while (!m_bAbort && (nErrResult = SQLGetDiagRec(fHandleType,
handle, sMsgNum++,
		szErrState, &dwErrCode, szErrText,
SQL_MAX_MESSAGE_LENGTH-1, &wErrMsgLen))
		!= SQL_NO_DATA)
	{
		if (!SQL_SUCCEEDED(nErrResult))
			break;

		if (m_pOutputFile && IsServerMessage(dwErrCode,
szErrText))
		{
			wsprintf(szBuffer, SM_SQLMSG_FORMAT,
(LPSTR)szErrText);
			temp = szBuffer;
			m_pOutputFile->WriteLine((BSTR) temp);
		}
		else if (IsODBCWarning(szErrState) && dwErrCode !=
SM_SQL_ERR_CHANGED_DB && dwErrCode != SM_SQL_ERR_CHANGED_LANG)
		{
			// Suppress warnings - 'Changed database context
to...' and 'Changed language setting to...'
			wsprintf(szBuffer, SM_SQLMSG_FORMAT,
ParseOdbcMsgPrefixes((LPCSTR)szErrText));
			temp = szBuffer;
			m_pOutputFile->WriteLine((BSTR) temp);
		}
		else if (m_pErrorFile && !IsODBCWarning(szErrState))
		{
			wsprintf(szBuffer, SM_SQLERR_FORMAT,
(LPSTR)szErrState, dwErrCode, (LPSTR)szErrText);
			temp = szBuffer;
			m_pErrorFile->WriteLine((BSTR) temp);
		}
	}
}

/*
    FUNCTION: LogErrors(SQLRETURN rc, SWORD fHandleType, SQLHANDLE
handle)
    COMMENTS: Writes the error message to the error log
*/

void CExecError::LogErrors(CExecute *p)
{
	_bstr_t temp(m_szExecErrorDesc[m_iErrCode]);

	if (p->m_pErrorFile)
		p->m_pErrorFile->WriteLine((BSTR)temp);

	return;
}
```

```
/*
    FUNCTION: ODBCCleanup(HDBC *hdbc, HSTMT *hstmt)
    COMMENTS: Cleanup of all ODBC structures
*/

void CExecute::ODBCCleanup(HDBC *hdbc, HSTMT *hstmt)
{
	SQLRETURN	lReturn;

#ifdef _DEBUG
	_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
ODBCCleanup.\n");
#endif

	if (*hstmt != SQL_NULL_HSTMT)
	{
#ifdef _DEBUG
		_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLCloseCursor.\n");
#endif
		SQLCloseCursor(hstmt);
#ifdef _DEBUG
		_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLFreeHandle for hstmt.\n");
#endif
		SQLFreeHandle(SQL_HANDLE_STMT, hstmt);
		*hstmt = SQL_NULL_HSTMT;
	}

	// Cleanup connection if it is a dynamic connection
	if (IsDynamicConnection())
	{
		if (*hdbc != SQL_NULL_HDBC)
		{
#ifdef _DEBUG
			_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLDisconnect.\n");
#endif
			lReturn = SQLDisconnect(*hdbc);
#ifdef _DEBUG
			_CrtDbgReport(_CRT_WARN, NULL, 0, NULL, "Executing
SQLFreeHandle for hdbc.\n");
#endif
			SQLFreeHandle(SQL_HANDLE_DBC, hdbc);
			*hdbc = SQL_NULL_HDBC;
		}
	}
	else
		ResetConnectionUsage();

	return;
}

// Wrapper function that raises an error if a Windows Api fails
STDMETHODIMP CExecute::RaiseSystemError(void)
```

```
{
        char s[MAXBUFLEN];

        GetSystemError(s);
        return Error(s, 0, NULL, GUID_NULL);
}

// Wrapper function that logs the error raised by an Api function to the
passed in file
void CExecute::LogSystemError(ISMLog *pFile)
{
        if (pFile)
        {
                char s[MAXBUFLEN];
                GetSystemError(s);

                _bstr_t temp(s);
                pFile->WriteLine((BSTR)temp);
        }
}

// Populates the passed in string with the last Windows Api error that
occurred
void CExecute::GetSystemError(LPSTR s)
{
        long c;
        DWORD e;

        e = GetLastError();

        c = sprintf(s, "Error code: %ld. ", e);
        c = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS,
                NULL, e, 0, s + c, MAXBUFLEN - c, NULL);

        return;
}

STDMETHODIMP CExecute::get_StepStatus(InstanceStatus *pVal)
{
        *pVal = m_StepStatus;
        return S_OK;
}

STDMETHODIMP CExecute::WriteError(BSTR szMsg)
{
        if (m_pErrorFile)
                return(m_pErrorFile->WriteLine(szMsg));

        return S_OK;
}
```

Execute.h

```
//  FILE:       Execute.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Declaration of the CExecute
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// Execute.h : Declaration of the CExecute

#ifndef __EXECUTE_H_
#define __EXECUTE_H_

#include <atlwin.h>
#include <comdef.h>
#include <stdio.h>
#include "resource.h"       // main symbols
#include "ExecuteDllCP.h"
#include "..\LogWriter\LogWriter.h"
#include "..\LogWriter\SMLog.h"
#include "..\common\SMTime\SMTime.h"
#include "..\common\SMTime\SMTimer.h"

// ODBC-specific includes
#define DBNTWIN32
#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>

////////////////////////////////////////////////////////////////////////
/////
// CExecute

#define WM_TASK_START  (WM_USER + 101)
#define WM_TASK_FINISH  (WM_USER + 102)

#define SM_SQLERR_FORMAT             "SQL Error State:%s, Native Error
Code: %ld\r\nODBC Error: %s"

                                // format for ODBC error messages
#define SM_SQLWARN_FORMAT            SM_SQLERR_FORMAT       // format
for ODBC warnings
#define SM_SQLMSG_FORMAT             "%s"                           //
format for messages from the server

#define SM_SQL_STATE_WARNING  "01000"
#define SM_MSG_SERVER                "[Microsoft][ODBC SQL Server
Driver][SQL Server]"

#define SM_SQL_ERR_CHANGED_DB 5701
#define SM_SQL_ERR_CHANGED_LANG      5703

#define SM_STEPMASTER_ERROR          "StepMaster Error: "

#define CMD_SEPARATOR                "\nGO"
```

```
#define INV_ARRAY_INDEX     -1                      // invalid index
into an array

#define MAXBUFLEN           256         // display buffer size
#define MAXLOGCMDLEN        256         // maximum characters in command
that will be

       // printed to log
#define MAXLOGCMDBUF        512         // maximum characters in command
that will be

       // printed to log
#define MAX_DATA_LEN        4000         // maximum buffer size for
variable-length data types

       // viz. character and binary fields
#define FILE_ACCESS_READ    "r"                     // Open file for
read access

#define S_NULL "NULL"

// Define a macro to increase the size of a buffer so it is a multiple
of teh alignment size.
// Thus, if a buffer starts on an alignment boundary, it will end just
before the next
// alignment boundary. Here, an alignment size of 4 is used because this
is the size of the
// largest data type used in the application's buffer - the size of an
SDWORD and of the largest
// default C data type are both 4. If a larger data type (such as
__int64) is used, it will be
// necessary to align for that size.
#define ALIGNSIZE 4
#define ALIGNBUF(Length) ((Length) % ALIGNSIZE) ? \
    ((Length) + ALIGNSIZE - ((Length) % ALIGNSIZE)) : (Length)

#define MAX_BUFFER_SIZE              64000

typedef enum OdbcOperations
{
       SMSQLAllocHandle,
       SMSQLDriverConnect,
       SMSQLExecDirect,
       SMSQLSetStmtAttr,
       SMSQLCancel,
       SMSQLNumResultCols,
       SMSQLDescribeCol,
       SMSQLColAttribute,
       SMSQLFetch,
       SMSQLGetData,
       SMSQLRowCount,
       SMSQLMoreResults,
       SMSQLBindCol,
};

class CODBCError
{
public:
       CODBCError(SQLRETURN nResult, SWORD fHandleType, SQLHANDLE
       handle, OdbcOperations FailedOp)
       {
              m_fHandleType  = fHandleType;
              m_handle              = handle;
              m_FailedOp            = FailedOp;
              m_nResult             = nResult;
       };

private:
       SWORD                 m_fHandleType;
       SQLHANDLE             m_handle;
       OdbcOperations m_FailedOp;
       SQLRETURN             m_nResult;

private:
inline BOOL IsServerMessage(SQLINTEGER lNativeError, UCHAR *szErr){
       return( (strstr((LPCTSTR)szErr, SM_MSG_SERVER) != NULL) ?
(lNativeError == 0) : FALSE); }
inline BOOL IsODBCWarning(UCHAR *szSqlState){
       return(strcmp((LPCSTR)szSqlState, SM_SQL_STATE_WARNING) == 0);}
inline LPCSTR ParseOdbcMsgPrefixes(LPCSTR szMsg){ char *pDest;
       return( (pDest = strstr(szMsg, SM_MSG_SERVER)) == NULL ? szMsg :
pDest + strlen(SM_MSG_SERVER));}
};

class ATL_NO_VTABLE CExecute :
       public CWindowImpl<CExecute>,
       public CComObjectRootEx<CComSingleThreadModel>,
       public CComCoClass<CExecute, &CLSID_Execute>,
       public IConnectionPointContainerImpl<CExecute>,
       public ISupportErrorInfo,
       public IDispatchImpl<IExecute, &IID_IExecute,
&LIBID_EXECUTEDLLLib>,
       public CProxy_IExecuteEvents< CExecute >
{
public:
       CExecute()
       {
              m_pErrorFile = NULL;
              //m_pLogFile = NULL;
              m_pOutputFile = NULL;

              // Initialize the elapsed time for the step
              m_tElapsedTime = 0;

              // Initialize the run status for the step
              m_StepStatus = gintPending;

              m_hHandle = SQL_NULL_HSTMT;
              m_bAbort = FALSE;

              m_iConnectionIndex = INV_ARRAY_INDEX;
       }
```

```
        ~CExecute()
        {
        }

        friend class CExecError;

public:
        DECLARE_WND_CLASS("Execute")

                BEGIN_MSG_MAP(CExecute)
                MESSAGE_HANDLER(WM_TASK_FINISH, OnTaskFinished)
                MESSAGE_HANDLER(WM_TASK_START, OnTaskStarted)
                END_MSG_MAP()
public:

        LRESULT OnTaskStarted(UINT uMsg, WPARAM wParam,
                LPARAM lParam, BOOL& bHandled)
        {
                CURRENCY       CStartTime = Get64BitTime(&m_tStartTime);

                Fire_Start(CStartTime);
                return 0;
        }

        LRESULT OnTaskFinished(UINT uMsg, WPARAM wParam,
                LPARAM lParam, BOOL& bHandled)
        {
                CURRENCY       CEndTime = Get64BitTime(&m_tEndTime);

                Fire_Complete(CEndTime, (long)m_tElapsedTime);
                return 0;
        }

        HRESULT FinalConstruct()
        {
                HRESULT        hr;
                RECT           rect;

                rect.left=0;
                rect.right=100;
                rect.top=0;
                rect.bottom=100;

                HWND hwnd = Create( NULL, rect, "ExecuteWindow",
WS_POPUP);

                if (!hwnd)
                        return HRESULT_FROM_WIN32(GetLastError());

                hr = CoCreateInstance(CLSID_SMLog, NULL, CLSCTX_INPROC,
                        IID_ISMLog, (void **)&m_pErrorFile);
                if FAILED(hr)
                        return(hr);
                m_pErrorFile->put_Append(TRUE);

                //hr = CoCreateInstance(CLSID_SMLog, NULL, CLSCTX_INPROC,
                //      IID_ISMLog, (void **)&m_pLogFile);
                //if FAILED(hr)
                //      return(hr);

                hr = CoCreateInstance(CLSID_SMLog, NULL, CLSCTX_INPROC,
                        IID_ISMLog, (void **)&m_pOutputFile);
                if FAILED(hr)
                        return(hr);
                m_pOutputFile->put_Append(TRUE);

                hr = CoCreateInstance(CLSID_SMTimer, NULL, CLSCTX_INPROC,
                        IID_ISMTimer, (void **)&m_ExecTime);
                if FAILED(hr)
                        return(hr);

                return S_OK;
        }

        void FinalRelease()
        {
                if (m_hWnd != NULL)
                        DestroyWindow();

                // Close the log and error files
                if (m_pErrorFile)
                        m_pErrorFile->Release();
                m_pErrorFile = NULL;

                //if (m_pLogFile)
                //      m_pLogFile->Release();
                //m_pLogFile = NULL;

                if (m_ExecTime)
                        m_ExecTime->Release();
                m_ExecTime = NULL;
        }

DECLARE_REGISTRY_RESOURCEID(IDR_EXECUTE)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CExecute)
        COM_INTERFACE_ENTRY(IExecute)
        COM_INTERFACE_ENTRY(ISupportErrorInfo)
        COM_INTERFACE_ENTRY(IDispatch)
        COM_INTERFACE_ENTRY(IConnectionPointContainer)
        COM_INTERFACE_ENTRY_IMPL(IConnectionPointContainer)
END_COM_MAP()
BEGIN_CONNECTION_POINT_MAP(CExecute)
CONNECTION_POINT_ENTRY(DIID__IExecuteEvents)
END_CONNECTION_POINT_MAP()

// ISupportsErrorInfo
STDMETHOD(InterfaceSupportsErrorInfo)(REFIID riid);
```

```
// IExecute
public:
	STDMETHOD(put_ErrorFile)(/*[in]*/ BSTR newVal);
	STDMETHOD(put_OutputFile)(/*[in]*/ BSTR newVal);
	STDMETHOD(WriteError)(BSTR szMsg);
	STDMETHOD(Abort)();
	STDMETHOD(get_StepStatus)(/*[out, retval]*/ InstanceStatus
*pVal);
	STDMETHOD(DoExecute)(/*[in]*/ BSTR szCommand, /*[in]*/ BSTR
szExecutionDtls, /*[in]*/ ExecutionType ExecMethod,
		/*[in]*/ BOOL bNoCount, /*[in]*/ BOOL bNoExecute,
/*[in]*/ BOOL bParseOnly,
		/*[in]*/ BOOL bQuotedIds, /*[in]*/ BOOL bAnsiNulls,
/*[in]*/ BOOL bShowQP,
		/*[in]*/ BOOL bStatsTime, /*[in]*/ BOOL bStatsIO,
/*[in]*/ long lRowCount,
		/*[in]*/ long lQueryTmout, /*[in]*/ BSTR szConnection);

	TC_TIME		ExecuteShell();
	TC_TIME		ExecuteODBC();
	STDMETHODIMP	AbortShell();
	STDMETHODIMP	AbortODBC();

	_bstr_t		m_szCommand;
	_bstr_t		m_szExecDtls;
	_bstr_t		m_szConnection;
	DWORD		m_lMode;
	//DATE		m_CurTime;
	SYSTEMTIME	m_tStartTime;
	SYSTEMTIME	m_tEndTime;
	TC_TIME		m_tElapsedTime;
	ISMLog		*m_pErrorFile;
	//ILog		*m_pLogFile;
	ISMLog		*m_pOutputFile;
	ISMTimer	*m_ExecTime;
	ExecutionType	m_ExecMthd;
	InstanceStatus m_StepStatus;
	HANDLE		m_hHandle;		// Process handle
for shell commands and
                                                                //
Statement handle for ODBC commands
	LPSTR		m_szCmd;

private:
	LPSTR		NextCmdInBatch(LPSTR szBatch);
	void		ProcessResultsets();
	SQLSMALLINT	GetDefaultCType(SQLINTEGER SQLType);
	void		PrintData(void *vData, SQLSMALLINT CType,
SQLINTEGER IndPtr, SQLSMALLINT iScale, ISMLog *pOutput);
	void		LogODBCErrors(SQLRETURN nResult, SWORD
fHandleType, SQLHANDLE handle, OdbcOperations FailedOp);
	void		ODBCCleanup(HDBC *hdbc, HSTMT *hstmt);
	STDMETHODIMP	RaiseSystemError(void);
	void		LogSystemError(ISMLog *pFile);
	void		GetSystemError(LPSTR s);
	void		SetConnectionOption(LPSTR szConn, HDBC
*pHdbc);
	void		ResetConnectionProperties(HDBC *p_hdbc);

	void		InitializeConnection(HDBC *phdbc, BOOL
*pbDoConnect);
	void		ReConnectDeadConnection(HDBC *phdbc);

	void		ResetConnectionUsage();

	int			m_iConnectionIndex;

	BOOL		m_bNoCount, m_bNoExecute, m_bParseOnly,
m_bQuotedIds, m_bAnsiNulls, \
			m_bShowQP, m_bStatsTime,
m_bStatsIO;
	long		m_lRowCount;
	SQLUINTEGER	m_lQueryTmout;
	_bstr_t		m_ErrorFile, m_OutputFile;
	BOOL		m_bAbort;

private:
inline BOOL IsServerMessage(SQLINTEGER lNativeError, UCHAR *szErr){
	return( (strstr((LPCTSTR)szErr, SM_MSG_SERVER) != NULL) ?
(lNativeError == 0) : FALSE); }
inline BOOL IsODBCWarning(UCHAR *szSqlState){
	return(strcmp((LPCSTR)szSqlState, SM_SQL_STATE_WARNING) == 0);}
inline BOOL IsErrorReturn(SQLRETURN iRetCode){
	return( (!SQL_SUCCEEDED(iRetCode)) && (iRetCode !=
SQL_NO_DATA) );}
inline LPCSTR ParseOdbcMsgPrefixes(LPCSTR szMsg){ char *pDest;
	return( (pDest = strstr(szMsg, SM_MSG_SERVER)) == NULL ? szMsg :
pDest + strlen(SM_MSG_SERVER));}
inline BOOL IsDynamicConnection(){ return(!strcmp((LPSTR)m_szConnection,
""));}
inline void HandleODBCError(SQLRETURN rc, SWORD fHandleType, SQLHANDLE
handle, OdbcOperations OdbcOp)
{
	if (rc != SQL_SUCCESS)
    {
		LogODBCErrors(rc, fHandleType, handle, OdbcOp);
		if (IsErrorReturn(rc))
			throw new CODBCError(rc, fHandleType, handle,
OdbcOp);
    }
	return;
}

};

class CExecError
{
public:
	typedef enum ExecErrorCodes
	{
		SM_ERR_CONN_IN_USE,
```

```
        };

        CExecError(int iError)
        {
                m_iErrCode      = iError;
        };

        void                    LogErrors(CExecute *p);

private:
        int                             m_iErrCode;
        static char             *m_szExecErrorDesc[];
};

void                    ExecutionThread(LPVOID lpParameter);

#ifdef _TPCH_AUDIT
        void                    WriteFileToTpchLog(LPSTR szFile, LPSTR
szFmt);
        void                    WriteToTpchLog(char *szMsg);
#endif

#endif //__EXECUTE_H_
```

ExecuteDll.cpp

```
//  FILE:       ExecuteDll.cpp
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Implementation of DLL Exports.
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//


// Note: Proxy/Stub Information
//      To build a separate proxy/stub DLL,
//      run nmake -f ExecuteDllps.mk in the project directory.

#include "stdafx.h"
#include "resource.h"
#include <initguid.h>

#include "..\LogWriter\LogWriter.h"
#include "..\LogWriter\LogWriter_i.c"

#include "..\common\SMTime\SMTime.h"
#include "..\common\SMTime\SMTime_i.c"

#include "ExecuteDll.h"
#include "SMExecute.h"

#include "ExecuteDll_i.c"
#include "Execute.h"

CComModule _Module;

BEGIN_OBJECT_MAP(ObjectMap)
OBJECT_ENTRY(CLSID_Execute, CExecute)
END_OBJECT_MAP()

SQLHENV henv = NULL;
                // ODBC environment handle

static char szCaption[] = "StepMaster";
        // Message box caption

CRITICAL_SECTION hConnections;
                // Critical section to serialize access to available
connections
SM_Connection_Info      *p_Connections = NULL;                          //
Pointer to open connections
int                                     iConnectionCount = 0;
                // Number of open connections

#ifdef _TPCH_AUDIT
        FILE *pfLogFile = NULL;
                        // Log file containing timestamps
        CRITICAL_SECTION hLogFileWrite;
                // Critical section to serialize writes to log

        static char szFileOpenModeAppend[] = "a+";                      //
Log file open mode

        static char szEnvVarLogFile[] = "TPCH_LOG_FILE";        //
Environment variable - initialized to

                                        // log file name if timing
information

                                        // is to be logged
#endif

void ShowODBCErrors(SWORD fHandleType, SQLHANDLE handle);
void CloseOpenConnections();

////////////////////////////////////////////////////////////////////////
/////
// DLL Entry Point

extern "C"
BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID
/*lpReserved*/)
{

    if (dwReason == DLL_PROCESS_ATTACH)
    {
        _Module.Init(ObjectMap, hInstance, &LIBID_EXECUTEDLLLib);
        DisableThreadLibraryCalls(hInstance);
```

```
#ifdef _TPCH_AUDIT
                char szMsg[MAXBUFLEN];
                LPSTR szLogFileName = getenv(szEnvVarLogFile);

                if (szLogFileName == NULL)
                {
                        sprintf(szMsg, "The environment variable '%s' does
not exist. "
                                "Step timing information will not be
written to a log.", szEnvVarLogFile);
                        MessageBox(NULL, szMsg, szCaption, MB_OK);
                }
                else
                {
                        if ( (pfLogFile = fopen(szLogFileName,
szFileOpenModeAppend)) == NULL )
                        {
                                sprintf(szMsg, "The file '%s' does not
exist. "
                                        "Step timing information will not
be written to log.", szLogFileName);
                                MessageBox(NULL, szMsg, szCaption, MB_OK);
                        }
                        else
                                InitializeCriticalSection(&hLogFileWrite);
                }
#endif

                InitializeCriticalSection(&hConnections);

                p_Connections = NULL;
                iConnectionCount = 0;

                if (!SQL_SUCCEEDED(SQLSetEnvAttr(NULL,
SQL_ATTR_CONNECTION_POOLING, (SQLPOINTER)SQL_CP_ONE_PER_HENV , 0)))
                        ShowODBCErrors(SQL_HANDLE_ENV, henv);

                if (!SQL_SUCCEEDED(SQLAllocHandle(SQL_HANDLE_ENV,
SQL_NULL_HANDLE, &henv)))
                {
                        ShowODBCErrors(SQL_HANDLE_ENV, henv);
                        return FALSE;
                }

                if (!SQL_SUCCEEDED(SQLSetEnvAttr(henv,
SQL_ATTR_ODBC_VERSION, (LPVOID)SQL_OV_ODBC3, 0)))
                        ShowODBCErrors(SQL_HANDLE_ENV, henv);

                SQLUINTEGER CpMatch;
                if (!SQL_SUCCEEDED(SQLGetEnvAttr(henv, SQL_ATTR_CP_MATCH,
&CpMatch, 0, NULL)))
                        ShowODBCErrors(SQL_HANDLE_ENV, henv);

                if (!SQL_SUCCEEDED(SQLSetEnvAttr(henv, SQL_ATTR_CP_MATCH,
(SQLPOINTER)SQL_CP_STRICT_MATCH, SQL_IS_INTEGER)))
                        ShowODBCErrors(SQL_HANDLE_ENV, henv);
        }
        else if (dwReason == DLL_PROCESS_DETACH)
            {

#ifdef _TPCH_AUDIT
                if (pfLogFile != NULL)
                {
                        fclose(pfLogFile);

                        DeleteCriticalSection(&hLogFileWrite);
                }
#endif

                CloseOpenConnections();

                if (henv != NULL)
                        SQLFreeEnv(henv);

                DeleteCriticalSection(&hConnections);

                _Module.Term();
            }
        return TRUE;    // ok
}

void ShowODBCErrors(SWORD fHandleType, SQLHANDLE handle)
{
        UCHAR           szErrState[SQL_SQLSTATE_SIZE+1];             //
SQL Error State string
        UCHAR           szErrText[SQL_MAX_MESSAGE_LENGTH+1];    // SQL
Error Text string
        char
        szBuffer[SQL_SQLSTATE_SIZE+SQL_MAX_MESSAGE_LENGTH+MAXBUFLEN+1] =
"";
                                        // formatted Error text Buffer
        SWORD           wErrMsgLen;
                        // Error message length
        SQLINTEGER      dwErrCode;
                        // Native Error code
        SQLRETURN       nErrResult;
                        // Return Code from SQLGetDiagRec
        SWORD           sMsgNum = 1;
                // Error sequence number

        // call SQLGetDiagRec function with proper ODBC handles,
repeatedly until
        // function returns SQL_NO_DATA.
       while ((nErrResult = SQLGetDiagRec(fHandleType, handle, sMsgNum++,
                szErrState, &dwErrCode, szErrText,
SQL_MAX_MESSAGE_LENGTH-1, &wErrMsgLen))
                != SQL_NO_DATA)
        {
                if (!SQL_SUCCEEDED(nErrResult))
                        break;
```

```
                wsprintf(szBuffer, SM_SQLERR_FORMAT, (LPSTR)szErrState,
dwErrCode, (LPSTR)szErrText);

                MessageBox(NULL, szBuffer, szCaption, MB_OK);
        }
}

void CloseOpenConnections()
{
        // Closes all open connections

        if (p_Connections)
        {
                for (int iConnIndex = iConnectionCount - 1; iConnIndex >=
0; iConnIndex--)
                {
                        if ((p_Connections + iConnIndex)->hdbc !=
SQL_NULL_HDBC)
                        {
#ifdef _DEBUG
                                _CrtDbgReport(_CRT_WARN, NULL, 0, NULL,
"Executing SQLDisconnect.\n");
#endif
                                SQLDisconnect((p_Connections +
iConnIndex)->hdbc);
#ifdef _DEBUG
                                _CrtDbgReport(_CRT_WARN, NULL, 0, NULL,
"Executing SQLFreeHandle for hdbc.\n");
#endif
                                SQLFreeHandle(SQL_HANDLE_DBC,
(p_Connections + iConnIndex)->hdbc );
                                (p_Connections + iConnIndex)->hdbc =
SQL_NULL_HDBC;
                        }

                }

                free(p_Connections);
        }
        p_Connections = NULL;

        return;
}

////////////////////////////////////////////////////////////////////////
/////
// Used to determine whether the DLL can be unloaded by OLE

STDAPI DllCanUnloadNow(void)
{
    return (_Module.GetLockCount()==0) ? S_OK : S_FALSE;
}

////////////////////////////////////////////////////////////////////////
/////
// Returns a class factory to create an object of the requested type

STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, LPVOID* ppv)
{
    return _Module.GetClassObject(rclsid, riid, ppv);
}

////////////////////////////////////////////////////////////////////////
/////
// DllRegisterServer - Adds entries to the system registry

STDAPI DllRegisterServer(void)
{
    // registers object, typelib and all interfaces in typelib
    return _Module.RegisterServer(TRUE);
}

////////////////////////////////////////////////////////////////////////
/////
// DllUnregisterServer - Removes entries from the system registry

STDAPI DllUnregisterServer(void)
{
    return _Module.UnregisterServer(TRUE);
}
```

ExecuteDll.h

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


/* File created by MIDL compiler version 5.01.0164 */
/* at Mon Jun 09 19:33:03 2003
 */
/* Compiler settings for
C:\charles\Stepmaster\ExecuteDll\ExecuteDll.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32, ms_ext, c_ext
    error checks: allocation ref bounds_check enum stub_data
*/
//@@MIDL_FILE_HEADING(  )


/* verify that the <rpcndr.h> version is high enough to compile this
file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 440
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __ExecuteDll_h__
#define __ExecuteDll_h__

#ifdef __cplusplus
```

```
extern "C"{
#endif

/* Forward Declarations */

#ifndef ___IExecuteEvents_FWD_DEFINED__
#define ___IExecuteEvents_FWD_DEFINED__
typedef interface _IExecuteEvents _IExecuteEvents;
#endif  /* ___IExecuteEvents_FWD_DEFINED__ */


#ifndef __IExecute_FWD_DEFINED__
#define __IExecute_FWD_DEFINED__
typedef interface IExecute IExecute;
#endif  /* __IExecute_FWD_DEFINED__ */


#ifndef __Execute_FWD_DEFINED__
#define __Execute_FWD_DEFINED__

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

#endif  /* __Execute_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"

void __RPC_FAR * __RPC_USER MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void __RPC_FAR * );

/* interface __MIDL_itf_ExecuteDll_0000 */
/* [local] */

typedef /* [helpstring][uuid] */
enum ExecutionType
    {   execODBC       = 0x1,
        execShell      = 0x2
    }   ExecutionType;

typedef /* [helpstring][uuid] */
enum InstanceStatus
    {   gintDisabled   = 0x1,
        gintPending    = 0x2,
        gintRunning    = 0x3,
        gintComplete   = 0x4,
        gintFailed     = 0x5,
        gintAborted    = 0x6
    }   InstanceStatus;

extern RPC_IF_HANDLE __MIDL_itf_ExecuteDll_0000_v0_0_c_ifspec;
extern RPC_IF_HANDLE __MIDL_itf_ExecuteDll_0000_v0_0_s_ifspec;


#ifndef __EXECUTEDLLLib_LIBRARY_DEFINED__
#define __EXECUTEDLLLib_LIBRARY_DEFINED__

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


EXTERN_C const IID LIBID_EXECUTEDLLLib;

#ifndef ___IExecuteEvents_DISPINTERFACE_DEFINED__
#define ___IExecuteEvents_DISPINTERFACE_DEFINED__

/* dispinterface _IExecuteEvents */
/* [helpstring][uuid] */


EXTERN_C const IID DIID__IExecuteEvents;

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

    MIDL_INTERFACE("551AC532-AB1C-11D2-BC0C-00A0C90D2CA5")
    _IExecuteEvents : public IDispatch
    {
    };

#else   /* C style interface */

    typedef struct _IExecuteEventsVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *QueryInterface )(
            _IExecuteEvents __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void __RPC_FAR *__RPC_FAR *ppvObject);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *AddRef )(
            _IExecuteEvents __RPC_FAR * This);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *Release )(
            _IExecuteEvents __RPC_FAR * This);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfoCount )(
            _IExecuteEvents __RPC_FAR * This,
            /* [out] */ UINT __RPC_FAR *pctinfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfo )(
            _IExecuteEvents __RPC_FAR * This,
            /* [in] */ UINT iTInfo,
            /* [in] */ LCID lcid,
            /* [out] */ ITypeInfo __RPC_FAR *__RPC_FAR *ppTInfo);
```

```
        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetIDsOfNames )(
            _IExecuteEvents __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [size_is][in] */ LPOLESTR __RPC_FAR *rgszNames,
            /* [in] */ UINT cNames,
            /* [in] */ LCID lcid,
            /* [size_is][out] */ DISPID __RPC_FAR *rgDispId);

        /* [local] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *Invoke )(
            _IExecuteEvents __RPC_FAR * This,
            /* [in] */ DISPID dispIdMember,
            /* [in] */ REFIID riid,
            /* [in] */ LCID lcid,
            /* [in] */ WORD wFlags,
            /* [out][in] */ DISPPARAMS __RPC_FAR *pDispParams,
            /* [out] */ VARIANT __RPC_FAR *pVarResult,
            /* [out] */ EXCEPINFO __RPC_FAR *pExcepInfo,
            /* [out] */ UINT __RPC_FAR *puArgErr);

        END_INTERFACE
    } _IExecuteEventsVtbl;

    interface _IExecuteEvents
    {
        CONST_VTBL struct _IExecuteEventsVtbl __RPC_FAR *lpVtbl;
    };



#ifdef COBJMACROS


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

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

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


#define _IExecuteEvents_GetTypeInfoCount(This,pctinfo)      \
    (This)->lpVtbl -> GetTypeInfoCount(This,pctinfo)

#define _IExecuteEvents_GetTypeInfo(This,iTInfo,lcid,ppTInfo)       \
    (This)->lpVtbl -> GetTypeInfo(This,iTInfo,lcid,ppTInfo)

#define
_IExecuteEvents_GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)
       \
    (This)->lpVtbl ->
GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)

#define
_IExecuteEvents_Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pV
arResult,pExcepInfo,puArgErr) \
    (This)->lpVtbl ->
Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResult,pExcepI
nfo,puArgErr)

#endif /* COBJMACROS */


#endif  /* C style interface */


#endif  /* ___IExecuteEvents_DISPINTERFACE_DEFINED__ */


#ifndef __IExecute_INTERFACE_DEFINED__
#define __IExecute_INTERFACE_DEFINED__

/* interface IExecute */
/* [unique][helpstring][dual][uuid][object] */


EXTERN_C const IID IID_IExecute;

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

    MIDL_INTERFACE("551AC531-AB1C-11D2-BC0C-00A0C90D2CA5")
    IExecute : public IDispatch
    {
    public:
        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
DoExecute(
            /* [in] */ BSTR szCommand,
            /* [in] */ BSTR szExecutionDtls,
            /* [in] */ ExecutionType ExecMethod,
            /* [in] */ BOOL bNoCount,
            /* [in] */ BOOL bNoExecute,
            /* [in] */ BOOL bParseOnly,
            /* [in] */ BOOL bQuotedIds,
            /* [in] */ BOOL bAnsiNulls,
            /* [in] */ BOOL bShowQP,
            /* [in] */ BOOL bStatsTime,
            /* [in] */ BOOL bStatsIO,
            /* [in] */ long lRowCount,
            /* [in] */ long lQueryTmout,
            /* [in] */ BSTR szConnection) = 0;

        virtual /* [helpstring][id][propget] */ HRESULT
STDMETHODCALLTYPE get_StepStatus(
            /* [retval][out] */ InstanceStatus __RPC_FAR *pVal) = 0;

        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
Abort( void) = 0;
```

```
        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
WriteError(
            BSTR szMsg) = 0;

        virtual /* [helpstring][id][propput] */ HRESULT
STDMETHODCALLTYPE put_OutputFile(
            /* [in] */ BSTR newVal) = 0;

        virtual /* [helpstring][id][propput] */ HRESULT
STDMETHODCALLTYPE put_ErrorFile(
            /* [in] */ BSTR newVal) = 0;

    };

#else   /* C style interface */

    typedef struct IExecuteVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *QueryInterface )(
            IExecute __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void __RPC_FAR *__RPC_FAR *ppvObject);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *AddRef )(
            IExecute __RPC_FAR * This);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *Release )(
            IExecute __RPC_FAR * This);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfoCount )(
            IExecute __RPC_FAR * This,
            /* [out] */ UINT __RPC_FAR *pctinfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfo )(
            IExecute __RPC_FAR * This,
            /* [in] */ UINT iTInfo,
            /* [in] */ LCID lcid,
            /* [out] */ ITypeInfo __RPC_FAR *__RPC_FAR *ppTInfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetIDsOfNames )(
            IExecute __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [size_is][in] */ LPOLESTR __RPC_FAR *rgszNames,
            /* [in] */ UINT cNames,
            /* [in] */ LCID lcid,
            /* [size_is][out] */ DISPID __RPC_FAR *rgDispId);

        /* [local] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *Invoke )(
            IExecute __RPC_FAR * This,
            /* [in] */ DISPID dispIdMember,
            /* [in] */ REFIID riid,
            /* [in] */ LCID lcid,
            /* [in] */ WORD wFlags,
            /* [out][in] */ DISPPARAMS __RPC_FAR *pDispParams,
            /* [out] */ VARIANT __RPC_FAR *pVarResult,
            /* [out] */ EXCEPINFO __RPC_FAR *pExcepInfo,
            /* [out] */ UINT __RPC_FAR *puArgErr);

        /* [helpstring][id] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR
*DoExecute )(
            IExecute __RPC_FAR * This,
            /* [in] */ BSTR szCommand,
            /* [in] */ BSTR szExecutionDtls,
            /* [in] */ ExecutionType ExecMethod,
            /* [in] */ BOOL bNoCount,
            /* [in] */ BOOL bNoExecute,
            /* [in] */ BOOL bParseOnly,
            /* [in] */ BOOL bQuotedIds,
            /* [in] */ BOOL bAnsiNulls,
            /* [in] */ BOOL bShowQP,
            /* [in] */ BOOL bStatsTime,
            /* [in] */ BOOL bStatsIO,
            /* [in] */ long lRowCount,
            /* [in] */ long lQueryTmout,
            /* [in] */ BSTR szConnection);

        /* [helpstring][id][propget] */ HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *get_StepStatus )(
            IExecute __RPC_FAR * This,
            /* [retval][out] */ InstanceStatus __RPC_FAR *pVal);

        /* [helpstring][id] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR
*Abort )(
            IExecute __RPC_FAR * This);

        /* [helpstring][id] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR
*WriteError )(
            IExecute __RPC_FAR * This,
            BSTR szMsg);

        /* [helpstring][id][propput] */ HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *put_OutputFile )(
            IExecute __RPC_FAR * This,
            /* [in] */ BSTR newVal);

        /* [helpstring][id][propput] */ HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *put_ErrorFile )(
            IExecute __RPC_FAR * This,
            /* [in] */ BSTR newVal);

        END_INTERFACE
    } IExecuteVtbl;

    interface IExecute
    {
        CONST_VTBL struct IExecuteVtbl __RPC_FAR *lpVtbl;
    };
```

```
#ifdef COBJMACROS


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

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

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


#define IExecute_GetTypeInfoCount(This,pctinfo)     \
    (This)->lpVtbl -> GetTypeInfoCount(This,pctinfo)

#define IExecute_GetTypeInfo(This,iTInfo,lcid,ppTInfo)      \
    (This)->lpVtbl -> GetTypeInfo(This,iTInfo,lcid,ppTInfo)

#define IExecute_GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)
       \
    (This)->lpVtbl ->
GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)

#define
IExecute_Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResul
t,pExcepInfo,puArgErr) \
    (This)->lpVtbl ->
Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResult,pExcepI
nfo,puArgErr)


#define
IExecute_DoExecute(This,szCommand,szExecutionDtls,ExecMethod,bNoCount,bN
oExecute,bParseOnly,bQuotedIds,bAnsiNulls,bShowQP,bStatsTime,bStatsIO,lR
owCount,lQueryTmout,szConnection)    \
    (This)->lpVtbl ->
DoExecute(This,szCommand,szExecutionDtls,ExecMethod,bNoCount,bNoExecute,
bParseOnly,bQuotedIds,bAnsiNulls,bShowQP,bStatsTime,bStatsIO,lRowCount,l
QueryTmout,szConnection)

#define IExecute_get_StepStatus(This,pVal)   \
    (This)->lpVtbl -> get_StepStatus(This,pVal)

#define IExecute_Abort(This)  \
    (This)->lpVtbl -> Abort(This)

#define IExecute_WriteError(This,szMsg)      \
    (This)->lpVtbl -> WriteError(This,szMsg)

#define IExecute_put_OutputFile(This,newVal) \
    (This)->lpVtbl -> put_OutputFile(This,newVal)

#define IExecute_put_ErrorFile(This,newVal)  \
    (This)->lpVtbl -> put_ErrorFile(This,newVal)

#endif /* COBJMACROS */


#endif  /* C style interface */


/* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
IExecute_DoExecute_Proxy(
    IExecute __RPC_FAR * This,
    /* [in] */ BSTR szCommand,
    /* [in] */ BSTR szExecutionDtls,
    /* [in] */ ExecutionType ExecMethod,
    /* [in] */ BOOL bNoCount,
    /* [in] */ BOOL bNoExecute,
    /* [in] */ BOOL bParseOnly,
    /* [in] */ BOOL bQuotedIds,
    /* [in] */ BOOL bAnsiNulls,
    /* [in] */ BOOL bShowQP,
    /* [in] */ BOOL bStatsTime,
    /* [in] */ BOOL bStatsIO,
    /* [in] */ long lRowCount,
    /* [in] */ long lQueryTmout,
    /* [in] */ BSTR szConnection);


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


/* [helpstring][id][propget] */ HRESULT STDMETHODCALLTYPE
IExecute_get_StepStatus_Proxy(
    IExecute __RPC_FAR * This,
    /* [retval][out] */ InstanceStatus __RPC_FAR *pVal);


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


/* [helpstring][id] */ HRESULT STDMETHODCALLTYPE IExecute_Abort_Proxy(
    IExecute __RPC_FAR * This);


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

```
/* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
IExecute_WriteError_Proxy(
    IExecute __RPC_FAR * This,
    BSTR szMsg);


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


/* [helpstring][id][propput] */ HRESULT STDMETHODCALLTYPE
IExecute_put_OutputFile_Proxy(
    IExecute __RPC_FAR * This,
    /* [in] */ BSTR newVal);


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


/* [helpstring][id][propput] */ HRESULT STDMETHODCALLTYPE
IExecute_put_ErrorFile_Proxy(
    IExecute __RPC_FAR * This,
    /* [in] */ BSTR newVal);


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



#endif  /* __IExecute_INTERFACE_DEFINED__ */


EXTERN_C const CLSID CLSID_Execute;

#ifdef __cplusplus

class DECLSPEC_UUID("2EFC198E-AA8D-11D2-BC0C-00A0C90D2CA5")
Execute;
#endif
#endif /* __EXECUTEDLLLib_LIBRARY_DEFINED__ */

/* Additional Prototypes for ALL interfaces */

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

ExecuteDllCP.h

```
//  FILE:       ExecuteDllCP.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Connection point implementation
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
#ifndef _EXECUTEDLLCP_H_
#define _EXECUTEDLLCP_H_



template <class T>
class CProxy_IExecuteEvents : public IConnectionPointImpl<T,
&DIID__IExecuteEvents, CComDynamicUnkArray>
{
        //Warning this class may be recreated by the wizard.
public:
        VOID Fire_Start(CY StartTime)
        {
                T* pT = static_cast<T*>(this);
                int nConnectionIndex;
                CComVariant* pvars = new CComVariant[1];
                int nConnections = m_vec.GetSize();

                for (nConnectionIndex = 0; nConnectionIndex <
nConnections; nConnectionIndex++)
                {
                        pT->Lock();
                        CComPtr<IUnknown> sp =
m_vec.GetAt(nConnectionIndex);
                        pT->Unlock();
                        IDispatch* pDispatch =
reinterpret_cast<IDispatch*>(sp.p);
                        if (pDispatch != NULL)
                        {
                                pvars[0] = StartTime;
                                DISPPARAMS disp = { pvars, NULL, 1, 0 };
                                pDispatch->Invoke(0x1, IID_NULL,
LOCALE_USER_DEFAULT, DISPATCH_METHOD, &disp, NULL, NULL, NULL);
                        }
```

```
                }
                delete[] pvars;

        }
        VOID Fire_Complete(CY EndTime, LONG Elapsed)
        {
                T* pT = static_cast<T*>(this);
                int nConnectionIndex;
                CComVariant* pvars = new CComVariant[2];
                int nConnections = m_vec.GetSize();

                for (nConnectionIndex = 0; nConnectionIndex <
nConnections; nConnectionIndex++)
                {
                        pT->Lock();
                        CComPtr<IUnknown> sp =
m_vec.GetAt(nConnectionIndex);
                        pT->Unlock();
                        IDispatch* pDispatch =
reinterpret_cast<IDispatch*>(sp.p);
                        if (pDispatch != NULL)
                        {
                                pvars[1] = EndTime;
                                pvars[0] = Elapsed;
                                DISPPARAMS disp = { pvars, NULL, 2, 0 };
                                pDispatch->Invoke(0x2, IID_NULL,
LOCALE_USER_DEFAULT, DISPATCH_METHOD, &disp, NULL, NULL, NULL);
                        }
                }
                delete[] pvars;

        }
};
#endif
```

resource.h

```
//  FILE:       resource.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Resource file
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by ExecuteDll.rc
//
#define IDS_PROJNAME                    100
#define IDR_EXECUTE                     101
#define IDR_LOG                         102
#define IDR_EXECUTESHELL                103

#define SM_SYSTEM_ERROR                 0x0201

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        201
#define _APS_NEXT_COMMAND_VALUE         32768
#define _APS_NEXT_CONTROL_VALUE         201
#define _APS_NEXT_SYMED_VALUE           104
#endif
#endif
```

SMExecute.h

```
//  FILE:       SMExecute.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Common include file for Execute.cpp and ExecuteDll.cpp
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//

#pragma once

// ODBC-specific includes
#define DBNTWIN32
#include <sqltypes.h>
#include <sql.h>
#include <sqlext.h>

#define CONNECTION_NAME_LEN 256         // connection name length

typedef struct _SM_Connection_Info
{
        char    szConnectionName[CONNECTION_NAME_LEN];
        HDBC    hdbc;
        BOOL    bInUse;
} SM_Connection_Info;
```

StdAfx.cpp

```
//  FILE:       stdafx.cpp
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    source file that includes just the standard includes
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// stdafx.cpp : source file that includes just the standard includes
//  stdafx.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information
```

```
#include "stdafx.h"

#ifdef _ATL_STATIC_REGISTRY
#include <statreg.h>
#include <statreg.cpp>
#endif

#include <atlimpl.cpp>
```

StdAfx.h

```
//  FILE:       stdafx.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    include file for standard system include files,
//      or project specific include files that are used frequently,
//      but are changed infrequently
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//

#if !defined(AFX_STDAFX_H__551AC528_AB1C_11D2_BC0C_00A0C90D2CA5__INCLUDE
D_)
#define AFX_STDAFX_H__551AC528_AB1C_11D2_BC0C_00A0C90D2CA5__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define STRICT
#ifndef _WIN32_WINNT
#define _WIN32_WINNT 0x0400
#endif
#define _ATL_APARTMENT_THREADED

#include <atlbase.h>
//You may derive a class from CComModule and use it if you want to
override
//something, but do not change the name of _Module
extern CComModule _Module;
#include <atlcom.h>

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif
// !defined(AFX_STDAFX_H__551AC528_AB1C_11D2_BC0C_00A0C90D2CA5__INCLUDED
)
```

F.1.2 LogWriter

LogWriter.cpp

```
//  FILE:       LogWriter.cpp
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Implementation of DLL Exports.
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// LogWriter.cpp : Implementation of DLL Exports.


// Note: Proxy/Stub Information
//      To build a separate proxy/stub DLL,
//      run nmake -f LogWriterps.mk in the project directory.

#include "stdafx.h"
#include "resource.h"
#include <initguid.h>
#include "LogWriter.h"

#include "LogWriter_i.c"
#include "SMLog.h"


CComModule _Module;

BEGIN_OBJECT_MAP(ObjectMap)
OBJECT_ENTRY(CLSID_SMLog, CSMLog)
END_OBJECT_MAP()

////////////////////////////////////////////////////////////////////////
/////
// DLL Entry Point

extern "C"
BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID
/*lpReserved*/)
{
    if (dwReason == DLL_PROCESS_ATTACH)
    {
        _Module.Init(ObjectMap, hInstance, &LIBID_LOGWRITERLib);
        DisableThreadLibraryCalls(hInstance);
    }
    else if (dwReason == DLL_PROCESS_DETACH)
        _Module.Term();
    return TRUE;    // ok
}

////////////////////////////////////////////////////////////////////////
/////
// Used to determine whether the DLL can be unloaded by OLE

STDAPI DllCanUnloadNow(void)
```

```
{
    return (_Module.GetLockCount()==0) ? S_OK : S_FALSE;
}

/////////////////////////////////////////////////////////////////////////////
/////
// Returns a class factory to create an object of the requested type

STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, LPVOID* ppv)
{
    return _Module.GetClassObject(rclsid, riid, ppv);
}

/////////////////////////////////////////////////////////////////////////////
/////
// DllRegisterServer - Adds entries to the system registry

STDAPI DllRegisterServer(void)
{
    // registers object, typelib and all interfaces in typelib
    return _Module.RegisterServer(TRUE);
}

/////////////////////////////////////////////////////////////////////////////
/////
// DllUnregisterServer - Removes entries from the system registry

STDAPI DllUnregisterServer(void)
{
    return _Module.UnregisterServer(TRUE);
}
```

LogWriter.h

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


/* File created by MIDL compiler version 5.01.0164 */
/* at Thu Sep 19 17:49:50 2002
 */
/* Compiler settings for C:\charles\Stepmaster\LogWriter\LogWriter.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32, ms_ext, c_ext
    error checks: allocation ref bounds_check enum stub_data
*/
//@@MIDL_FILE_HEADING(  )


/* verify that the <rpcndr.h> version is high enough to compile this
file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 440
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __RPCNDR_H_VERSION__
#error this stub requires an updated version of <rpcndr.h>
#endif // __RPCNDR_H_VERSION__

#ifndef COM_NO_WINDOWS_H
#include "windows.h"
#include "ole2.h"
#endif /*COM_NO_WINDOWS_H*/

#ifndef __LogWriter_h__
#define __LogWriter_h__

#ifdef __cplusplus
extern "C"{
#endif

/* Forward Declarations */

#ifndef __ISMLog_FWD_DEFINED__
#define __ISMLog_FWD_DEFINED__
typedef interface ISMLog ISMLog;
#endif  /* __ISMLog_FWD_DEFINED__ */


#ifndef ___ISMLogEvents_FWD_DEFINED__
#define ___ISMLogEvents_FWD_DEFINED__
typedef interface _ISMLogEvents _ISMLogEvents;
#endif  /* ___ISMLogEvents_FWD_DEFINED__ */


#ifndef __SMLog_FWD_DEFINED__
#define __SMLog_FWD_DEFINED__

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

#endif  /* __SMLog_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"

void __RPC_FAR * __RPC_USER MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void __RPC_FAR * );

#ifndef __ISMLog_INTERFACE_DEFINED__
#define __ISMLog_INTERFACE_DEFINED__

/* interface ISMLog */
/* [unique][helpstring][dual][uuid][object] */
```

```
EXTERN_C const IID IID_ISMLog;

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

    MIDL_INTERFACE("5AC75DAD-1936-11D3-BC2D-00A0C90D2CA5")
    ISMLog : public IDispatch
    {
    public:
        virtual /* [helpstring][id][propput] */ HRESULT
STDMETHODCALLTYPE put_FileHeader(
            /* [in] */ BSTR newVal) = 0;

        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
WriteLine(
            BSTR szMsg) = 0;

        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
WriteField(
            BSTR szMsg) = 0;

        virtual /* [helpstring][id][propput] */ HRESULT
STDMETHODCALLTYPE put_FileName(
            /* [in] */ BSTR newVal) = 0;

        virtual /* [helpstring][id][propput] */ HRESULT
STDMETHODCALLTYPE put_Append(
            /* [in] */ BOOL newVal) = 0;

    };

#else   /* C style interface */

    typedef struct ISMLogVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *QueryInterface )(
            ISMLog __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void __RPC_FAR *__RPC_FAR *ppvObject);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *AddRef )(
            ISMLog __RPC_FAR * This);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *Release )(
            ISMLog __RPC_FAR * This);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfoCount )(
            ISMLog __RPC_FAR * This,
            /* [out] */ UINT __RPC_FAR *pctinfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfo )(
            ISMLog __RPC_FAR * This,
            /* [in] */ UINT iTInfo,
            /* [in] */ LCID lcid,
            /* [out] */ ITypeInfo __RPC_FAR *__RPC_FAR *ppTInfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetIDsOfNames )(
            ISMLog __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [size_is][in] */ LPOLESTR __RPC_FAR *rgszNames,
            /* [in] */ UINT cNames,
            /* [in] */ LCID lcid,
            /* [size_is][out] */ DISPID __RPC_FAR *rgDispId);

        /* [local] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *Invoke )(
            ISMLog __RPC_FAR * This,
            /* [in] */ DISPID dispIdMember,
            /* [in] */ REFIID riid,
            /* [in] */ LCID lcid,
            /* [in] */ WORD wFlags,
            /* [out][in] */ DISPPARAMS __RPC_FAR *pDispParams,
            /* [out] */ VARIANT __RPC_FAR *pVarResult,
            /* [out] */ EXCEPINFO __RPC_FAR *pExcepInfo,
            /* [out] */ UINT __RPC_FAR *puArgErr);

        /* [helpstring][id][propput] */ HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *put_FileHeader )(
            ISMLog __RPC_FAR * This,
            /* [in] */ BSTR newVal);

        /* [helpstring][id] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR
*WriteLine )(
            ISMLog __RPC_FAR * This,
            BSTR szMsg);

        /* [helpstring][id] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR
*WriteField )(
            ISMLog __RPC_FAR * This,
            BSTR szMsg);

        /* [helpstring][id][propput] */ HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *put_FileName )(
            ISMLog __RPC_FAR * This,
            /* [in] */ BSTR newVal);

        /* [helpstring][id][propput] */ HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *put_Append )(
            ISMLog __RPC_FAR * This,
            /* [in] */ BOOL newVal);

        END_INTERFACE
    } ISMLogVtbl;

    interface ISMLog
    {
        CONST_VTBL struct ISMLogVtbl __RPC_FAR *lpVtbl;
    };
```

```
#ifdef COBJMACROS


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

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

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


#define ISMLog_GetTypeInfoCount(This,pctinfo)        \
    (This)->lpVtbl -> GetTypeInfoCount(This,pctinfo)

#define ISMLog_GetTypeInfo(This,iTInfo,lcid,ppTInfo) \
    (This)->lpVtbl -> GetTypeInfo(This,iTInfo,lcid,ppTInfo)

#define ISMLog_GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)
       \
    (This)->lpVtbl ->
GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)

#define
ISMLog_Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResult,
pExcepInfo,puArgErr)   \
    (This)->lpVtbl ->
Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResult,pExcepI
nfo,puArgErr)


#define ISMLog_put_FileHeader(This,newVal)    \
    (This)->lpVtbl -> put_FileHeader(This,newVal)

#define ISMLog_WriteLine(This,szMsg)  \
    (This)->lpVtbl -> WriteLine(This,szMsg)

#define ISMLog_WriteField(This,szMsg) \
    (This)->lpVtbl -> WriteField(This,szMsg)

#define ISMLog_put_FileName(This,newVal)     \
    (This)->lpVtbl -> put_FileName(This,newVal)

#define ISMLog_put_Append(This,newVal)       \
    (This)->lpVtbl -> put_Append(This,newVal)

#endif /* COBJMACROS */


#endif  /* C style interface */



/* [helpstring][id][propput] */ HRESULT STDMETHODCALLTYPE
ISMLog_put_FileHeader_Proxy(
    ISMLog __RPC_FAR * This,
    /* [in] */ BSTR newVal);


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


/* [helpstring][id] */ HRESULT STDMETHODCALLTYPE ISMLog_WriteLine_Proxy(
    ISMLog __RPC_FAR * This,
    BSTR szMsg);


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


/* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
ISMLog_WriteField_Proxy(
    ISMLog __RPC_FAR * This,
    BSTR szMsg);


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


/* [helpstring][id][propput] */ HRESULT STDMETHODCALLTYPE
ISMLog_put_FileName_Proxy(
    ISMLog __RPC_FAR * This,
    /* [in] */ BSTR newVal);


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


/* [helpstring][id][propput] */ HRESULT STDMETHODCALLTYPE
ISMLog_put_Append_Proxy(
    ISMLog __RPC_FAR * This,
    /* [in] */ BOOL newVal);
```

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



#endif  /* __ISMLog_INTERFACE_DEFINED__ */



#ifndef __LOGWRITERLib_LIBRARY_DEFINED__
#define __LOGWRITERLib_LIBRARY_DEFINED__

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


EXTERN_C const IID LIBID_LOGWRITERLib;

#ifndef ___ISMLogEvents_DISPINTERFACE_DEFINED__
#define ___ISMLogEvents_DISPINTERFACE_DEFINED__

/* dispinterface _ISMLogEvents */
/* [helpstring][uuid] */


EXTERN_C const IID DIID__ISMLogEvents;

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

    MIDL_INTERFACE("5AC75DB1-1936-11D3-BC2D-00A0C90D2CA5")
    _ISMLogEvents : public IDispatch
    {
    };

#else   /* C style interface */

    typedef struct _ISMLogEventsVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *QueryInterface )(
            _ISMLogEvents __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void __RPC_FAR *__RPC_FAR *ppvObject);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *AddRef )(
            _ISMLogEvents __RPC_FAR * This);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *Release )(
            _ISMLogEvents __RPC_FAR * This);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfoCount )(
            _ISMLogEvents __RPC_FAR * This,
            /* [out] */ UINT __RPC_FAR *pctinfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfo )(
            _ISMLogEvents __RPC_FAR * This,
            /* [in] */ UINT iTInfo,
            /* [in] */ LCID lcid,
            /* [out] */ ITypeInfo __RPC_FAR *__RPC_FAR *ppTInfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetIDsOfNames )(
            _ISMLogEvents __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [size_is][in] */ LPOLESTR __RPC_FAR *rgszNames,
            /* [in] */ UINT cNames,
            /* [in] */ LCID lcid,
            /* [size_is][out] */ DISPID __RPC_FAR *rgDispId);

        /* [local] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *Invoke )(
            _ISMLogEvents __RPC_FAR * This,
            /* [in] */ DISPID dispIdMember,
            /* [in] */ REFIID riid,
            /* [in] */ LCID lcid,
            /* [in] */ WORD wFlags,
            /* [out][in] */ DISPPARAMS __RPC_FAR *pDispParams,
            /* [out] */ VARIANT __RPC_FAR *pVarResult,
            /* [out] */ EXCEPINFO __RPC_FAR *pExcepInfo,
            /* [out] */ UINT __RPC_FAR *puArgErr);

        END_INTERFACE
    } _ISMLogEventsVtbl;

    interface _ISMLogEvents
    {
        CONST_VTBL struct _ISMLogEventsVtbl __RPC_FAR *lpVtbl;
    };



#ifdef COBJMACROS


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

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

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


#define _ISMLogEvents_GetTypeInfoCount(This,pctinfo) \
    (This)->lpVtbl -> GetTypeInfoCount(This,pctinfo)

#define _ISMLogEvents_GetTypeInfo(This,iTInfo,lcid,ppTInfo) \
    (This)->lpVtbl -> GetTypeInfo(This,iTInfo,lcid,ppTInfo)
```

```
#define
_ISMLogEvents_GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)
        \
    (This)->lpVtbl ->
GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)

#define
_ISMLogEvents_Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVar
Result,pExcepInfo,puArgErr)    \
    (This)->lpVtbl ->
Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResult,pExcepI
nfo,puArgErr)

#endif /* COBJMACROS */


#endif  /* C style interface */


#endif  /* ___ISMLogEvents_DISPINTERFACE_DEFINED__ */


EXTERN_C const CLSID CLSID_SMLog;

#ifdef __cplusplus

class DECLSPEC_UUID("5AC75DB0-1936-11D3-BC2D-00A0C90D2CA5")
SMLog;
#endif
#endif /* __LOGWRITERLib_LIBRARY_DEFINED__ */

/* Additional Prototypes for ALL interfaces */

unsigned long             __RPC_USER  BSTR_UserSize(     unsigned long
__RPC_FAR *, unsigned long            , BSTR __RPC_FAR * );
unsigned char __RPC_FAR * __RPC_USER  BSTR_UserMarshal(  unsigned long
__RPC_FAR *, unsigned char __RPC_FAR *, BSTR __RPC_FAR * );
unsigned char __RPC_FAR * __RPC_USER  BSTR_UserUnmarshal(unsigned long
__RPC_FAR *, unsigned char __RPC_FAR *, BSTR __RPC_FAR * );
void                      __RPC_USER  BSTR_UserFree(     unsigned long
__RPC_FAR *, BSTR __RPC_FAR * );

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

LogWriterCP.h

```
//  FILE:       LogWriterCP.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Connection point implementation
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
#ifndef _LOGWRITERCP_H_
#define _LOGWRITERCP_H_

template <class T>
class CProxy_ISMLogEvents : public IConnectionPointImpl<T,
&DIID__ISMLogEvents, CComDynamicUnkArray>
{
        //Warning this class may be recreated by the wizard.
public:
};
#endif
```

resource.h

```
//  FILE:       resource.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Resource file
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by LogWriter.rc
//
#define IDS_PROJNAME                    100
#define IDR_SMLOG                       101

// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        201
#define _APS_NEXT_COMMAND_VALUE         32768
#define _APS_NEXT_CONTROL_VALUE         201
#define _APS_NEXT_SYMED_VALUE           102
#endif
#endif
```

SMLog.cpp

```
//  FILE:       SMLog.cpp
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Implementation of CSMLog
```

```
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// SMLog.cpp : Implementation of CSMLog
#include "stdafx.h"
#include <stdio.h>
#include "LogWriter.h"
#include "SMLog.h"

/////////////////////////////////////////////////////////////////////////
/////
// CSMLog

STDMETHODIMP CSMLog::InterfaceSupportsErrorInfo(REFIID riid)
{
        static const IID* arr[] =
        {
                &IID_ISMLog
        };
        for (int i=0; i < sizeof(arr) / sizeof(arr[0]); i++)
        {
                if (InlineIsEqualGUID(*arr[i],riid))
                        return S_OK;
        }
        return S_FALSE;
}

STDMETHODIMP CSMLog::WriteToFile(BSTR szMsg)
{
    // Writes the passed in string to the file
        _bstr_t szTempMsg(szMsg);

        return(Write((PBYTE)(LPSTR)szTempMsg, SysStringLen(szMsg)));
}

HRESULT CSMLog::Init()
{
        char    szDrive[256];
        char    szDir[256];
        char    szLogDir[256];
        HANDLE  hLogThread;
        DWORD   dwThreadID;
        _bstr_t szFile(m_szFile);
        DWORD   lDisposition;

        //create transaction log directory
        _splitpath((LPCTSTR)szFile, szDrive, szDir, NULL, NULL);
        _makepath(szLogDir, szDrive, szDir, NULL, NULL);
        CreateDirectory(szLogDir, NULL);

    iBufferSize = WRITE_BUFFER_SIZE;
        iBytesFreeInBuffer     = iBufferSize;

        // use VirtualAlloc to get page aligned buffers //
        for (int i=0;i<MAX_NUM_BUFFERS;i++)
        {
                // use VirtualAlloc to get page aligned buffers //
                pBuffer[i] = (BYTE *)VirtualAlloc(NULL, iBufferSize,
MEM_COMMIT, PAGE_READWRITE );
                if (pBuffer[i] == NULL)
                        return RaiseSystemError();
        }

        iActiveBuffer = 0;
        pCurrent = pBuffer[iActiveBuffer];

        lDisposition = m_bAppend ? OPEN_ALWAYS : CREATE_ALWAYS;
        m_hTxnFile = CreateFile((LPCTSTR)szFile, GENERIC_WRITE,
FILE_SHARE_READ,
                NULL, lDisposition, FILE_ATTRIBUTE_NORMAL, NULL);
        if ( m_hTxnFile == INVALID_HANDLE_VALUE )
                return (RaiseSystemError());

        if (m_bAppend)
                if ( SetFilePointer(m_hTxnFile, 0, NULL, FILE_END) ==
ERR_SET_FILE_POINTER )
                        return (RaiseSystemError());

        hIoComplete  = CreateEvent(NULL, TRUE, TRUE, NULL);
        if ( hIoComplete == NULL)
                return RaiseSystemError();

        hLogFileIo = CreateEvent(NULL, FALSE, FALSE, NULL);
        if ( hLogFileIo == NULL)
                return RaiseSystemError();

        hLogThread = CreateThread( NULL, 0,
(LPTHREAD_START_ROUTINE)LogFileIO, this, 0, &dwThreadID );
        if (hLogThread == NULL)
                return RaiseSystemError();

        if (m_szHeader != NULL)
                WriteLine(m_szHeader);

        return S_OK;
}

void CSMLog::LogFileIO(void *ptr)
{
        unsigned long  BytesWritten;
        CSMLog *p=(CSMLog *)ptr;

        while( TRUE )
        {
                WaitForSingleObject(p->hLogFileIo, INFINITE);
                if ( p->m_hTxnFile == INVALID_HANDLE_VALUE )
                        break;

                // do synchronous (blocking) write to log file
                if ( !WriteFile(p->m_hTxnFile, p->pBuffer[p->iIoBuffer],
p->iWriteSize, &BytesWritten, NULL) )
                {
```

```
                        // set error code in this thread, but don't throw
an exception
                        // because no one will catch it.
                        p->dwError = GetLastError();

                }
                SetEvent(p->hIoComplete);
        }

        SetEvent(p->hIoComplete);
}

HRESULT CSMLog::Write(BYTE *ptr, DWORD iSize)
{
        int             StartPos, Remainder;
        int             dwErrorLocal = 0;

        if (!m_bInitialized)
        {
                HRESULT hr = Init();
                m_bInitialized = TRUE;

                if (FAILED(hr))
                        return hr;
        }

        if ( m_hTxnFile == INVALID_HANDLE_VALUE )
                return S_OK;

        if ( iBytesFreeInBuffer >= iSize )
        {
                memcpy(pCurrent, ptr, iSize);
                pCurrent += iSize;
                iBytesFreeInBuffer -= iSize;
        }
        else
        {
                // We don't expect to ever have to wait here, but just in
case...
                WaitForSingleObject(hIoComplete, INFINITE);

                // check for an error from the log writer thread
                if (dwError != 0)
                {
                        SetLastError(dwError);
                        return RaiseSystemError();
                }

                assert( iSize <= iBufferSize );
                memcpy(pCurrent, ptr, iBytesFreeInBuffer);
                StartPos = iBytesFreeInBuffer;
                Remainder = iSize - iBytesFreeInBuffer;

                // trigger an IO on the current buffer and roll to the
next buffer
                iIoBuffer = iActiveBuffer;
                iWriteSize = iBufferSize;
                ResetEvent(hIoComplete);
                SetEvent( hLogFileIo );                 // wake up IO writer

                iActiveBuffer = (iActiveBuffer+1) % MAX_NUM_BUFFERS;
                pCurrent = pBuffer[iActiveBuffer];

                memcpy(pCurrent, ((BYTE *)ptr+StartPos), Remainder);
                pCurrent += Remainder;
                iBytesFreeInBuffer = iBufferSize - Remainder;
        }

        return S_OK;
}

void CSMLog::CloseLogFile(void)
{
        if ( m_hTxnFile != INVALID_HANDLE_VALUE )
        {
                if ( iBytesFreeInBuffer < iBufferSize )
                {
                        WaitForSingleObject(hIoComplete, INFINITE);
                        ResetEvent(hIoComplete);

                        // check for an error from the log writer thread
                        if (dwError != 0)
                        {
                                SetLastError( dwError );
                                goto exit_SpinLock;
                        }

                        //zero fill remainder of buffer
                        ZeroMemory(pCurrent, iBytesFreeInBuffer);

                        iIoBuffer = iActiveBuffer;
                        iWriteSize = iBufferSize - iBytesFreeInBuffer;
                        SetEvent(hLogFileIo);           // wake up IO writer
                }

                WaitForSingleObject(hIoComplete, INFINITE);
                // check for an error from the log writer thread
                if (dwError != 0)
                        goto exit_SpinLock;

                pCurrent = pBuffer[iActiveBuffer];
                ZeroMemory(pCurrent, iBufferSize);
                iIoBuffer = iActiveBuffer;

                CloseHandle(m_hTxnFile);
                m_hTxnFile = INVALID_HANDLE_VALUE;              //handle to
open transaction log file

                // wake up IO writer one more time for it to terminate
                ResetEvent(hIoComplete);
                SetEvent(hLogFileIo);           // wake up IO writer
```

```
                WaitForSingleObject(hIoComplete, INFINITE);
        }

exit_SpinLock:

        if (dwError != 0)
        {
                if (m_hTxnFile != INVALID_HANDLE_VALUE)
                {
                        CloseHandle(m_hTxnFile);
                        m_hTxnFile = INVALID_HANDLE_VALUE;
                }

                SetLastError( dwError );
                // TODO: Don't know yet what to do with an error on the
file close,
                // since this function is called by the desctructor
(which does not return a value)
                //throw new CSystemErr( CSystemErr::eWriteFile,
"CTxnLog::CloseTransactionLogFile" );
        }

}

// Wrapper function that raises an error if a Windows Api fails
STDMETHODIMP CSMLog::RaiseSystemError(void)
{
        char s[ERR_BUFFER_SIZE];
        long c;
        DWORD e;

        e = GetLastError();

        c = sprintf(s, "Error code: %ld. ", e);
        c = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM |
FORMAT_MESSAGE_IGNORE_INSERTS,
                NULL, e, 0, s + c, sizeof(s) - c, NULL);

        return Error(s, 0, NULL, GUID_NULL);
}

//DEL STDMETHODIMP CSMLog::get_FileName(BSTR *pVal)
//DEL {
//DEL   *pVal = m_szFile;
//DEL   return S_OK;
//DEL }

STDMETHODIMP CSMLog::put_FileName(BSTR newVal)
{
        m_szFile = SysAllocString(newVal);

        return S_OK;
}

//DEL STDMETHODIMP CSMLog::get_FileHeader(BSTR *pVal)
//DEL {
//DEL   *pVal = m_szHeader;
//DEL   return S_OK;
//DEL }

STDMETHODIMP CSMLog::put_FileHeader(BSTR newVal)
{
        m_szHeader = SysAllocString(newVal);
        return S_OK;
}

STDMETHODIMP CSMLog::WriteLine(BSTR szMsg)
{
        _bstr_t szTmp(szMsg);

        szTmp += "\r";
        szTmp += "\n";
        return(WriteToFile(szTmp));
}

STDMETHODIMP CSMLog::WriteField(BSTR szMsg)
{
        return(WriteToFile(szMsg));
}

//DEL STDMETHODIMP CSMLog::get_Append(BOOL *pVal)
//DEL {
//DEL   *pVal = m_bAppend;
//DEL   return S_OK;
//DEL }

STDMETHODIMP CSMLog::put_Append(BOOL newVal)
{
        m_bAppend = newVal;
        return S_OK;
}
```

SMLog.h

```
//  FILE:       SMExecute.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Declaration of the CSMLog
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// SMLog.h : Declaration of the CSMLog

#ifndef __SMLOG_H_
#define __SMLOG_H_

#include "resource.h"       // main symbols
#include "assert.h"
#include "comdef.h"
#include "LogWriterCP.h"
```

```
#define WRITE_BUFFER_SIZE             8*1024
#define MAX_NUM_BUFFERS                       2
#define ERR_BUFFER_SIZE                       512
#define ERR_SET_FILE_POINTER  0xFFFFFFFF

////////////////////////////////////////////////////////////////////
/////
// CSMLog
class ATL_NO_VTABLE CSMLog :
        public CComObjectRootEx<CComSingleThreadModel>,
        public CComCoClass<CSMLog, &CLSID_SMLog>,
        public ISupportErrorInfo,
        public IConnectionPointContainerImpl<CSMLog>,
        public IDispatchImpl<ISMLog, &IID_ISMLog, &LIBID_LOGWRITERLib>,
        public CProxy_ISMLogEvents< CSMLog >
{
public:
        CSMLog()
        {
                m_bAppend = FALSE;
                m_bInitialized = FALSE;

         m_hTxnFile = INVALID_HANDLE_VALUE;
         hIoComplete = NULL;
         hLogFileIo = NULL;

                for (int i=0;i<MAX_NUM_BUFFERS;i++)
                        pBuffer[i] = NULL;
         pCurrent = NULL;

         m_szFile = NULL;
         m_szHeader = NULL;
         dwError = 0;
        }

    ~CSMLog()
    {
                if (m_hTxnFile != INVALID_HANDLE_VALUE)
                        CloseLogFile();

                for (int i=0;i<MAX_NUM_BUFFERS;i++)
                {
                        if (pBuffer[i] != NULL)
                                VirtualFree(pBuffer[i], 0, MEM_RELEASE);
                }

                if (m_szFile)
                        SysFreeString(m_szFile);
                m_szFile = NULL;

                if (m_szHeader)
                        SysFreeString(m_szHeader);
                m_szHeader = NULL;

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

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

        }

DECLARE_REGISTRY_RESOURCEID(IDR_SMLOG)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CSMLog)
        COM_INTERFACE_ENTRY(ISMLog)
        COM_INTERFACE_ENTRY(IDispatch)
        COM_INTERFACE_ENTRY(ISupportErrorInfo)
        COM_INTERFACE_ENTRY(IConnectionPointContainer)
        COM_INTERFACE_ENTRY_IMPL(IConnectionPointContainer)
END_COM_MAP()
BEGIN_CONNECTION_POINT_MAP(CSMLog)
CONNECTION_POINT_ENTRY(DIID__ISMLogEvents)
END_CONNECTION_POINT_MAP()


// ISupportsErrorInfo
        STDMETHOD(InterfaceSupportsErrorInfo)(REFIID riid);

// ISMLog
public:
        STDMETHOD(Init)();
        STDMETHOD(put_Append)(/*[in]*/ BOOL newVal);
        STDMETHOD(WriteField)(/*[in]*/ BSTR szMsg);
        STDMETHOD(WriteLine)(/*[in]*/ BSTR szMsg);
        STDMETHOD(put_FileHeader)(/*[in]*/ BSTR newVal);
        STDMETHOD(put_FileName)(/*[in]*/ BSTR newVal);

//      typedef enum err
private:

        BOOL            m_bAppend;
        BOOL            m_bInitialized;

        HANDLE          m_hTxnFile;                             //handle to
log file
        HANDLE          hIoComplete;                    //event to signify
that there are no pending IOs
        HANDLE          hLogFileIo;                             //event to
signal the IO thread to write the inactive buffer

        BYTE            *pBuffer[MAX_NUM_BUFFERS];
        BYTE            *pCurrent;                              //ptr to
current buffer
        int                     iActiveBuffer;          //indicates
which buffer is active: 0 or 1
        int                     iIoBuffer;
        //buffer for any pending IO operation
```

```
        DWORD           dwError;

        DWORD           iBufferSize;                    //buffer allocated
size
        DWORD           iBytesFreeInBuffer;             //total bytes
available for use in buffer
        DWORD           iWriteSize;                             //bytes to
write to file

        BSTR            m_szFile;                               //name of
the file to write to
        BSTR            m_szHeader;                             //header to
be printed to the file (optional)

private:
        static void             LogFileIO(void *p);
        HRESULT                 Write(BYTE *ptr, DWORD iSize);
        STDMETHODIMP    WriteToFile(BSTR szMsg);
        void                    CloseLogFile(void);
        STDMETHODIMP    RaiseSystemError(void);
};

#endif //__SMLOG_H_
```

StdAfx.cpp

```
//  FILE:       stdafx.cpp
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    source file that includes just the standard includes
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// stdafx.cpp : source file that includes just the standard includes
//  stdafx.pch will be the pre-compiled header
//  stdafx.obj will contain the pre-compiled type information

#include "stdafx.h"

#ifdef _ATL_STATIC_REGISTRY
#include <statreg.h>
#include <statreg.cpp>
#endif

#include <atlimpl.cpp>
```

StdAfx.h

```
//  FILE:       stdafx.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    include file for standard system include files,
//      or project specific include files that are used frequently,
//      but are changed infrequently
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//

#if !defined(AFX_STDAFX_H__5AC75DA4_1936_11D3_BC2D_00A0C90D2CA5__INCLUDE
D_)
#define AFX_STDAFX_H__5AC75DA4_1936_11D3_BC2D_00A0C90D2CA5__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define STRICT
#ifndef _WIN32_WINNT
#define _WIN32_WINNT 0x0400
#endif
#define _ATL_APARTMENT_THREADED

#include <atlbase.h>
//You may derive a class from CComModule and use it if you want to
override
//something, but do not change the name of _Module
extern CComModule _Module;
#include <atlcom.h>

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif
// !defined(AFX_STDAFX_H__5AC75DA4_1936_11D3_BC2D_00A0C90D2CA5__INCLUDED
)
```

F.1.3 SMTime

resource.h

```
//  FILE:       resource.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Resource file
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
//{{NO_DEPENDENCIES}}
// Microsoft Developer Studio generated include file.
// Used by SMTime.rc
//
#define IDS_PROJNAME                    100
#define IDR_SMTIMER                     102
```

```
// Next default values for new objects
//
#ifdef APSTUDIO_INVOKED
#ifndef APSTUDIO_READONLY_SYMBOLS
#define _APS_NEXT_RESOURCE_VALUE        201
#define _APS_NEXT_COMMAND_VALUE         32768
#define _APS_NEXT_CONTROL_VALUE         201
#define _APS_NEXT_SYMED_VALUE           103
#endif
#endif
```

SMTime.cpp

```
//  FILE:       SMTime.cpp
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Implementation of DLL Exports.
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// SMTime.cpp : Implementation of DLL Exports.


// Note: Proxy/Stub Information
//      To build a separate proxy/stub DLL,
//      run nmake -f SMTimeps.mk in the project directory.

#include "stdafx.h"
#include "resource.h"
#include <initguid.h>
#include "SMTime.h"

#include "SMTime_i.c"
#include "SMTimer.h"


CComModule _Module;

BEGIN_OBJECT_MAP(ObjectMap)
OBJECT_ENTRY(CLSID_SMTimer, CSMTimer)
END_OBJECT_MAP()

////////////////////////////////////////////////////////////////////////////
/////
// DLL Entry Point

extern "C"
BOOL WINAPI DllMain(HINSTANCE hInstance, DWORD dwReason, LPVOID
/*lpReserved*/)
{
    if (dwReason == DLL_PROCESS_ATTACH)
    {
        _Module.Init(ObjectMap, hInstance, &LIBID_SMTIMELib);
        DisableThreadLibraryCalls(hInstance);
    }
    else if (dwReason == DLL_PROCESS_DETACH)
        _Module.Term();
    return TRUE;    // ok
}

////////////////////////////////////////////////////////////////////////////
/////
// Used to determine whether the DLL can be unloaded by OLE

STDAPI DllCanUnloadNow(void)
{
    return (_Module.GetLockCount()==0) ? S_OK : S_FALSE;
}

////////////////////////////////////////////////////////////////////////////
/////
// Returns a class factory to create an object of the requested type

STDAPI DllGetClassObject(REFCLSID rclsid, REFIID riid, LPVOID* ppv)
{
    return _Module.GetClassObject(rclsid, riid, ppv);
}

////////////////////////////////////////////////////////////////////////////
/////
// DllRegisterServer - Adds entries to the system registry

STDAPI DllRegisterServer(void)
{
    // registers object, typelib and all interfaces in typelib
    return _Module.RegisterServer(TRUE);
}

////////////////////////////////////////////////////////////////////////////
/////
// DllUnregisterServer - Removes entries from the system registry

STDAPI DllUnregisterServer(void)
{
    return _Module.UnregisterServer(TRUE);
}
```

SMTime.h

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


/* File created by MIDL compiler version 5.01.0164 */
/* at Thu Sep 19 17:49:55 2002
 */
/* Compiler settings for C:\charles\Stepmaster\COMMON\SMTime\SMTime.idl:
    Oicf (OptLev=i2), W1, Zp8, env=Win32, ms_ext, c_ext
    error checks: allocation ref bounds_check enum stub_data
```

```
*/
//@@MIDL_FILE_HEADING(  )


/* verify that the <rpcndr.h> version is high enough to compile this
file*/
#ifndef __REQUIRED_RPCNDR_H_VERSION__
#define __REQUIRED_RPCNDR_H_VERSION__ 440
#endif

#include "rpc.h"
#include "rpcndr.h"

#ifndef __RPCNDR_H_VERSION__
#error this stub requires an updated version of <rpcndr.h>
#endif // __RPCNDR_H_VERSION__

#ifndef COM_NO_WINDOWS_H
#include "windows.h"
#include "ole2.h"
#endif /*COM_NO_WINDOWS_H*/

#ifndef __SMTime_h__
#define __SMTime_h__

#ifdef __cplusplus
extern "C"{
#endif

/* Forward Declarations */

#ifndef __ISMTimer_FWD_DEFINED__
#define __ISMTimer_FWD_DEFINED__
typedef interface ISMTimer ISMTimer;
#endif  /* __ISMTimer_FWD_DEFINED__ */


#ifndef __SMTimer_FWD_DEFINED__
#define __SMTimer_FWD_DEFINED__

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

#endif  /* __SMTimer_FWD_DEFINED__ */


/* header files for imported files */
#include "oaidl.h"
#include "ocidl.h"

void __RPC_FAR * __RPC_USER MIDL_user_allocate(size_t);
void __RPC_USER MIDL_user_free( void __RPC_FAR * );

#ifndef __ISMTimer_INTERFACE_DEFINED__
#define __ISMTimer_INTERFACE_DEFINED__

/* interface ISMTimer */
/* [unique][helpstring][dual][uuid][object] */


EXTERN_C const IID IID_ISMTimer;

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

    MIDL_INTERFACE("1A6D0AE4-8528-453B-B8E3-8DAD1F0561B7")
    ISMTimer : public IDispatch
    {
    public:
        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE
Start( void) = 0;

        virtual /* [helpstring][id] */ HRESULT STDMETHODCALLTYPE Stop(
            CURRENCY __RPC_FAR *pElapsedTime) = 0;

        virtual /* [helpstring][id][propget] */ HRESULT
STDMETHODCALLTYPE get_Running(
            /* [retval][out] */ BOOL __RPC_FAR *pVal) = 0;

    };

#else   /* C style interface */

    typedef struct ISMTimerVtbl
    {
        BEGIN_INTERFACE

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *QueryInterface )(
            ISMTimer __RPC_FAR * This,
            /* [in] */ REFIID riid,
            /* [iid_is][out] */ void __RPC_FAR *__RPC_FAR *ppvObject);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *AddRef )(
            ISMTimer __RPC_FAR * This);

        ULONG ( STDMETHODCALLTYPE __RPC_FAR *Release )(
            ISMTimer __RPC_FAR * This);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfoCount )(
            ISMTimer __RPC_FAR * This,
            /* [out] */ UINT __RPC_FAR *pctinfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetTypeInfo )(
            ISMTimer __RPC_FAR * This,
            /* [in] */ UINT iTInfo,
            /* [in] */ LCID lcid,
            /* [out] */ ITypeInfo __RPC_FAR *__RPC_FAR *ppTInfo);

        HRESULT ( STDMETHODCALLTYPE __RPC_FAR *GetIDsOfNames )(
            ISMTimer __RPC_FAR * This,
```

```
            /* [in] */ REFIID riid,
            /* [size_is][in] */ LPOLESTR __RPC_FAR *rgszNames,
            /* [in] */ UINT cNames,
            /* [in] */ LCID lcid,
            /* [size_is][out] */ DISPID __RPC_FAR *rgDispId);

        /* [local] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR *Invoke )(
            ISMTimer __RPC_FAR * This,
            /* [in] */ DISPID dispIdMember,
            /* [in] */ REFIID riid,
            /* [in] */ LCID lcid,
            /* [in] */ WORD wFlags,
            /* [out][in] */ DISPPARAMS __RPC_FAR *pDispParams,
            /* [out] */ VARIANT __RPC_FAR *pVarResult,
            /* [out] */ EXCEPINFO __RPC_FAR *pExcepInfo,
            /* [out] */ UINT __RPC_FAR *puArgErr);

        /* [helpstring][id] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR
*Start )(
            ISMTimer __RPC_FAR * This);

        /* [helpstring][id] */ HRESULT ( STDMETHODCALLTYPE __RPC_FAR
*Stop )(
            ISMTimer __RPC_FAR * This,
            CURRENCY __RPC_FAR *pElapsedTime);

        /* [helpstring][id][propget] */ HRESULT ( STDMETHODCALLTYPE
__RPC_FAR *get_Running )(
            ISMTimer __RPC_FAR * This,
            /* [retval][out] */ BOOL __RPC_FAR *pVal);

        END_INTERFACE
    } ISMTimerVtbl;

    interface ISMTimer
    {
        CONST_VTBL struct ISMTimerVtbl __RPC_FAR *lpVtbl;
    };



#ifdef COBJMACROS


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

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

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


#define ISMTimer_GetTypeInfoCount(This,pctinfo)     \
    (This)->lpVtbl -> GetTypeInfoCount(This,pctinfo)

#define ISMTimer_GetTypeInfo(This,iTInfo,lcid,ppTInfo)      \
    (This)->lpVtbl -> GetTypeInfo(This,iTInfo,lcid,ppTInfo)

#define ISMTimer_GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)
      \
    (This)->lpVtbl ->
GetIDsOfNames(This,riid,rgszNames,cNames,lcid,rgDispId)

#define
ISMTimer_Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResul
t,pExcepInfo,puArgErr) \
    (This)->lpVtbl ->
Invoke(This,dispIdMember,riid,lcid,wFlags,pDispParams,pVarResult,pExcepI
nfo,puArgErr)


#define ISMTimer_Start(This)  \
    (This)->lpVtbl -> Start(This)

#define ISMTimer_Stop(This,pElapsedTime)     \
    (This)->lpVtbl -> Stop(This,pElapsedTime)

#define ISMTimer_get_Running(This,pVal)      \
    (This)->lpVtbl -> get_Running(This,pVal)

#endif /* COBJMACROS */


#endif  /* C style interface */



/* [helpstring][id] */ HRESULT STDMETHODCALLTYPE ISMTimer_Start_Proxy(
    ISMTimer __RPC_FAR * This);


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


/* [helpstring][id] */ HRESULT STDMETHODCALLTYPE ISMTimer_Stop_Proxy(
    ISMTimer __RPC_FAR * This,
    CURRENCY __RPC_FAR *pElapsedTime);


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

```
/* [helpstring][id][propget] */ HRESULT STDMETHODCALLTYPE
ISMTimer_get_Running_Proxy(
    ISMTimer __RPC_FAR * This,
    /* [retval][out] */ BOOL __RPC_FAR *pVal);


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



#endif  /* __ISMTimer_INTERFACE_DEFINED__ */



#ifndef __SMTIMELib_LIBRARY_DEFINED__
#define __SMTIMELib_LIBRARY_DEFINED__

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


EXTERN_C const IID LIBID_SMTIMELib;


#ifndef __StepMasterTimeFunctions_MODULE_DEFINED__
#define __StepMasterTimeFunctions_MODULE_DEFINED__


/* module StepMasterTimeFunctions */
/* [dllname][version][helpstring] */

/* [entry][helpstring] */ CURRENCY __stdcall Get64BitTime(
    /* [in] */ LPSYSTEMTIME lpInitTime);

/* [entry][helpstring] */ void __stdcall JulianToTime(
    /* [in] */ CURRENCY julianTS,
    /* [out][in] */ int __RPC_FAR *yr,
    /* [out][in] */ int __RPC_FAR *mm,
    /* [out][in] */ int __RPC_FAR *dd,
    /* [out][in] */ int __RPC_FAR *hh,
    /* [out][in] */ int __RPC_FAR *mi,
    /* [out][in] */ int __RPC_FAR *ss,
    /* [out][in] */ int __RPC_FAR *ms);

#endif /* __StepMasterTimeFunctions_MODULE_DEFINED__ */

EXTERN_C const CLSID CLSID_SMTimer;

#ifdef __cplusplus

class DECLSPEC_UUID("27BAB71B-89E1-4A78-8854-FDFFBDC8037E")
SMTimer;
#endif
#endif /* __SMTIMELib_LIBRARY_DEFINED__ */

/* Additional Prototypes for ALL interfaces */

/* end of Additional Prototypes */

#ifdef __cplusplus
}
#endif

#endif
```

SMTimer.cpp

```
//  FILE:       SMTimer.cpp
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Implementation of CSMTimer
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// SMTimer.cpp : Implementation of CSMTimer
#include "stdafx.h"
#include "SMTime.h"
#include "SMTimer.h"

/////////////////////////////////////////////////////////////////////////////
/////
// CSMTimer

/////////////////////////////////////////////////////////////////////////////
// Construction/Destruction
/////////////////////////////////////////////////////////////////////////////

CSMTimer::~CSMTimer()
{

}

STDMETHODIMP CSMTimer::Start()
{
        // Starts the timer
        assert(!m_bInProcess);
        m_bInProcess = TRUE;

        m_lStartTime = MyTickCount();

        return S_OK;
}

STDMETHODIMP CSMTimer::Stop(CURRENCY *pElapsedTime)
{
        TC_TIME        lEndTime = MyTickCount();
```

```
	// Stops the timer and returns the elapsed time
	assert(m_bInProcess);
	m_bInProcess = FALSE;

	pElapsedTime->int64 = lEndTime - m_lStartTime;

	return S_OK;
}

TC_TIME CSMTimer::MyTickCount(void)
{
	TC_TIME currentTC;
	LARGE_INTEGER  l;
	__int64 count;

	//The purpose of this function is to prevent the 49 day wrapping
effect of the
	//system API GetTickCount(). This function essentially provides a
monotonically
	//increasing timer value which is milliseconds from class
instantiation.

	if ( m_bCountUnavailable )
	{
		count = (__int64)GetTickCount();
		currentTC = (TC_TIME)(count-m_baseTC);
	}
	else
	{
		QueryPerformanceCounter(&l);
		count = (__int64)l.HighPart << 32 | (__int64)l.LowPart;
		currentTC = (TC_TIME)(((count-m_baseTC) * 1000) /
m_Timerfreq);
	}

	return currentTC;
}

STDMETHODIMP CSMTimer::get_Running(BOOL *pVal)
{
	*pVal = m_bInProcess;

	return S_OK;
}

CURRENCY __stdcall Get64BitTime(LPSYSTEMTIME lpInitTime)
{
	__int64 ms_day, ms_hour, ms_minute, ms_seconds, ms_milliseconds,
ms_total;
	int		day;
	SYSTEMTIME	tim;
	CURRENCY	tmReturn;

	if ( lpInitTime )
		memcpy(&tim, lpInitTime, sizeof(SYSTEMTIME));
	else
		GetLocalTime(&tim);
	day = JulianDay((int)tim.wYear, (int)tim.wMonth, (int)tim.wDay);

	ms_day		= (__int64)day * (__int64)(24 * 1000 * 60
* 60);
	ms_hour		= (__int64)tim.wHour   * (__int64)(1000 *
3600);
	ms_minute	= (__int64)tim.wMinute * (1000 * 60);
	ms_seconds	= (__int64)(tim.wSecond * 1000);
	ms_milliseconds	= (__int64)tim.wMilliseconds;

	ms_total = ms_day + ms_hour + ms_minute + ms_seconds +
ms_milliseconds;
	tmReturn.int64 = ms_total;

	return tmReturn;
}

// JulianDay computes the number of days since Jan 1, 1900.
// This function is valid for dates from 1-Jan-1900 to 1-Jan-2100.
// 1-Jan-1900 = 0
int JulianDay( int yr, int mm, int dd )
{
	// MonthArray contains cumulative days for months in a non leap-
year
	int MonthArray[12] = { 0, 31, 59, 90, 120, 151, 181, 212, 243,
273, 304, 334};
	int j1, j2;

	// compute day of year (j1)
	j1 = MonthArray[mm-1] + dd - 1;
	// adjust day of year if this is a leap year and it is after
February
	if ((yr % 4)==0 && (yr != 1900) && (mm > 2))
		j1++;
	// compute number of days from 1/1/1900 to beginning of present
year
	j2 = (yr-1900)*365 + (yr-1901)/4;
	return j1+j2;

}

// Breaks up the Julian Time into it's sub-components
void __stdcall SMTime_JulianToTime( CURRENCY CurJulian, int* yr, int* mm,
int* dd, int *hh, int *mi, int *ss, int *ms )
{
	int julianDay, msLeft;
	JULIAN_TIME		julianTS = CurJulian.int64;

	*ms = julianTS % 1000;

	julianTS /= 1000;

	julianDay = (int)(julianTS / ( 60 * 60 * 24 ));
```

```
        JulianToCalendar(julianDay, yr, mm, dd );

        msLeft = (int)(julianTS - (julianDay * (__int64)( 60 * 60 *
24 )));

        *hh = msLeft / (60 * 60);
        msLeft = msLeft - *hh * 3600;
        *mi = msLeft / (60);
        *ss = msLeft % 60;
}

// JulianToCalendar converts a day index (from the JulianDay function)
to
// its corresponding calendar value (mm/dd/yr).  The valid range for
days
// is { 0 .. 73049 } for dates from 1-Jan-1900 to 1-Jan-2100.
void JulianToCalendar( int day, int* yr, int* mm, int* dd )
{
        int y, m, d;
        // month array contains days of months for months in a non leap-
year
        int month[12] = { 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30,
31 };

        // compute year from days
        if (day < 365)
                y = 1900;
        else
                y = 1901 + ((day-365)/1461)*4 + (4*((day-
365)%1461)+3)/1461;

        // adjust February if this year is a leap year
        if ((y % 4)==0 && (y != 1900))
                month[1] = 29;
        else
                month[1] = 28;

        d = day - JulianDay( y, 1, 1 ) + 1;
        m = 1;

        while (d > month[m-1])
        {
                d = d - month[m-1];
                m++;
        }

        *yr = y;
        *mm = m;
        *dd = d;
}
```

SMTimer.h

```
//  FILE:       SMTimer.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    Declaration of the CSMTimer
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//
// SMTimer.h : Declaration of the CSMTimer

#ifndef __SMTIMER_H_
#define __SMTIMER_H_

#include "resource.h"       // main symbols

#include "assert.h"

#define MAX_JULIAN_TIME              0x7FFFFFFFFFFFFFFF
#define JULIAN_TIME    __int64
#define TC_TIME        DWORD

#ifdef SMTIMER
#define DLL_LINK       __declspec( dllexport )
#else
#define DLL_LINK       __declspec( dllimport )
#endif

#ifdef __cplusplus
extern "C"
{
#endif
//DLL_LINK     CURRENCY       __stdcall SMTime_Get64BitTime(LPSYSTEMTIME
lpInitTime);
int                 JulianDay( int yr, int mm, int dd );
void           JulianToCalendar( int day, int* yr, int* mm, int* dd );
#ifdef __cplusplus
}
#endif


/////////////////////////////////////////////////////////////////////////
/////
// CSMTimer
class ATL_NO_VTABLE CSMTimer :
       public CComObjectRootEx<CComSingleThreadModel>,
       public CComCoClass<CSMTimer, &CLSID_SMTimer>,
       public IDispatchImpl<ISMTimer, &IID_ISMTimer, &LIBID_SMTIMELib>
{
public:
       CSMTimer()
       {
              LARGE_INTEGER  l;

              if ( !QueryPerformanceFrequency(&l) )
              {
                     m_baseTC = (__int64)GetTickCount();
                     m_bCountUnavailable = TRUE;
              }
```

```
		else
		{
			m_bCountUnavailable = FALSE;

			m_Timerfreq = (__int64)l.HighPart << 32 |
(__int64)l.LowPart;
			QueryPerformanceCounter(&l);
			m_baseTC = (__int64)l.HighPart << 32 |
(__int64)l.LowPart;
		}
		m_bInProcess = FALSE;
	}

DECLARE_REGISTRY_RESOURCEID(IDR_SMTIMER)

DECLARE_PROTECT_FINAL_CONSTRUCT()

BEGIN_COM_MAP(CSMTimer)
	COM_INTERFACE_ENTRY(ISMTimer)
	COM_INTERFACE_ENTRY(IDispatch)
//	COM_INTERFACE_ENTRY2(IDispatch, ISMTimer)
END_COM_MAP()

// ISMTimer
public:
	STDMETHOD(get_Running)(/*[out, retval]*/ BOOL *pVal);
	STDMETHOD(Stop)(CURRENCY *pElapsedTime);
	STDMETHOD(Start)();
	virtual ~CSMTimer();

private:
	__int64		m_baseTC;
	__int64		m_Timerfreq;
	BOOL		m_bCountUnavailable;
	TC_TIME		m_lStartTime;
	BOOL		m_bInProcess;

	TC_TIME		MyTickCount(void);
};

#endif //__SMTIMER_H_
```

StdAfx.cpp

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

#include "stdafx.h"

#ifdef _ATL_STATIC_REGISTRY
#include <statreg.h>
#include <statreg.cpp>
#endif

#include <atlimpl.cpp>
```

StdAfx.h

```
//  FILE:       stdafx.h
//              Microsoft TPC-H Kit Ver. 1.00
//              Copyright Microsoft, 1999
//              All Rights Reserved
//
//
//  PURPOSE:    include file for standard system include files,
//      or project specific include files that are used frequently,
//      but are changed infrequently
//  Contact:    Reshma Tharamal (reshmat@microsoft.com)
//

#if !defined(AFX_STDAFX_H__4CDF88F4_EE9C_4F29_8212_61557F251BDD__INCLUDE
D_)
#define AFX_STDAFX_H__4CDF88F4_EE9C_4F29_8212_61557F251BDD__INCLUDED_

#if _MSC_VER > 1000
#pragma once
#endif // _MSC_VER > 1000

#define STRICT
#ifndef _WIN32_WINNT
#define _WIN32_WINNT 0x0400
#endif
#define _ATL_APARTMENT_THREADED

#include <atlbase.h>
//You may derive a class from CComModule and use it if you want to
override
//something, but do not change the name of _Module
extern CComModule _Module;
#include <atlcom.h>

//{{AFX_INSERT_LOCATION}}
// Microsoft Visual C++ will insert additional declarations immediately
before the previous line.

#endif
// !defined(AFX_STDAFX_H__4CDF88F4_EE9C_4F29_8212_61557F251BDD__INCLUDED
)
```

F.1.4 StepMaster

cArrConstraints.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cArrConstraints"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
```

```
Attribute VB_Exposed = False
'  FILE:       cArrConstraints.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cConstraint objects.
'              Type-safe wrapper around cNodeCollections.
'              Also contains additional functions that determine all the
'              constraints for a step, all constraints in a workspace,
'              validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrConstraints As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cArrConstraints."
Public Sub SaveWspConstraints(ByVal lngWorkspace As Long)
    ' Calls a procedure to commit all changes to the constraints
    ' in the passed in workspace.

    Call mcarrConstraints.Save(lngWorkspace)

End Sub
Public Property Set ConstraintDB(vdata As Database)

    Set mcarrConstraints.NodeDB = vdata

End Property
Public Property Get ConstraintDB() As Database

    Set ConstraintDB = mcarrConstraints.NodeDB

End Property

Public Sub Modify(cConsToUpdate As cConstraint)

    ' Modify the constraint record
    Call mcarrConstraints.Modify(cConsToUpdate)

End Sub
Public Sub CreateNewConstraintVersion(ByVal lngStepId As Long, _
        ByVal strNewVersion As String, _
        ByVal strOldVersion As String, _
        ByVal intStepType As Integer)

    ' Does all the processing needed to create new versions of
    ' all the constraints for a given step
    ' It inserts new constraint records in the database with
    ' the new version numbers on them
    ' It also updates the version number on all constraints
    ' for the step in the array to the new version passed in
    ' Since it handles both global and manager/worker steps,
    ' it checks for the step_id or global_step_id fields,
    ' depending on the type of step

    Dim lngIndex As Long
    Dim cUpdateConstraint As cConstraint

    On Error GoTo CreateNewConstraintVersionErr
    mstrSource = mstrModuleName & "CreateNewConstraintVersion"

    ' Update the version/global version on Constraint with the
    ' passed in step/global step id
    For lngIndex = 0 To mcarrConstraints.Count - 1
        Set cUpdateConstraint = mcarrConstraints(lngIndex)
        If intStepType = gintGlobalStep Then
            If cUpdateConstraint.GlobalStepId = lngStepId And _
                    cUpdateConstraint.IndOperation <> DeleteOp Then
                cUpdateConstraint.GlobalVersionNo = strNewVersion

                ' Set the operation to indicate an insert
                cUpdateConstraint.IndOperation = InsertOp
            End If
        Else
            If cUpdateConstraint.StepId = lngStepId And _
                    cUpdateConstraint.IndOperation <> DeleteOp Then
                cUpdateConstraint.VersionNo = strNewVersion

                ' Set the operation to indicate an insert
                cUpdateConstraint.IndOperation = InsertOp
            End If
        End If
    Next lngIndex

    Exit Sub

CreateNewConstraintVersionErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "CreateNewConstraintVersion"
    On Error GoTo 0
    Err.Raise vbObjectError + errCreateNewConstraintVersionFailed, _
            mstrSource, _
            LoadResString(errCreateNewConstraintVersionFailed)

End Sub
Private Sub Class_Initialize()

    Set mcarrConstraints = New cNodeCollections
    BugMessage "cArrConstraints: Initialize event - setting Constraint
count to 0"

End Sub

Private Sub Class_Terminate()

    Set mcarrConstraints = Nothing
```

```
    BugMessage "cArrConstraints: Terminate event triggered"

End Sub

Public Sub Add(ByVal cConstraintToAdd As cConstraint)

    Set cConstraintToAdd.NodeDB = mcarrConstraints.NodeDB

    ' Retrieve a unique constraint identifier
    cConstraintToAdd.ConstraintId = cConstraintToAdd.NextIdentifier

    ' Call a procedure to load the constraint record in the array
    Call mcarrConstraints.Add(cConstraintToAdd)

End Sub
Public Sub Delete(ByVal cOldConstraint As cConstraint)

    Dim lngDeleteElement As Long
    Dim cConsToDelete As cConstraint

    lngDeleteElement = QueryConstraintIndex(cOldConstraint.ConstraintId)
    Set cConsToDelete = mcarrConstraints(lngDeleteElement)

    Call mcarrConstraints.Delete(cConsToDelete.Position)

    Set cConsToDelete = Nothing

End Sub
Private Function QueryConstraintIndex(lngConstraintId As Long) _
        As Long

    Dim lngIndex As Integer

    ' Find the element in the array to be deleted
    For lngIndex = 0 To mcarrConstraints.Count - 1

        ' Note: The constraint id is not a primary key field in
        ' the database - there can be multiple records with the
        ' same constraint_id but for different versions of a step
        ' However, since we'll always load the constraint information
        ' for the latest version of a step, we'll have just one
        ' constraint record with a given constraint_id
        If mcarrConstraints(lngIndex).ConstraintId = lngConstraintId
Then
            QueryConstraintIndex = lngIndex
            Exit Function
        End If

    Next lngIndex

    ' Raise error that Constraint has not been found
    ShowError errConstraintNotFound
    On Error GoTo 0
    Err.Raise vbObjectError + errConstraintNotFound, mstrSource, _
        LoadResString(errConstraintNotFound)

End Function

Public Function QueryConstraint(ByVal lngConstraintId As Long) _
    As cConstraint

    ' Returns a cConstraint object with the property values
    ' corresponding to the Constraint Identifier, lngConstraintId

    Dim lngQueryElement As Long

    lngQueryElement = QueryConstraintIndex(lngConstraintId)

    ' Set the return value to the queried Constraint
    Set QueryConstraint = mcarrConstraints(lngQueryElement)

End Function

Public Sub LoadConstraints(ByVal lngWorkspaceId As Long, rstStepsInWsp
As Recordset)

    ' Loads the constraints array with all the constraints
    ' for the workspace
    Dim recConstraints As Recordset
    Dim qyCons As DAO.QueryDef
    Dim strSql As String
    Dim dtStart As Date

    On Error GoTo LoadConstraintsErr
    mstrSource = mstrModuleName & "LoadConstraints"

    If rstStepsInWsp.RecordCount = 0 Then
        Exit Sub
    End If

    ' First check if the database object has been set
    If mcarrConstraints.NodeDB Is Nothing Then
        On Error GoTo 0
        Err.Raise vbObjectError + errSetDBBeforeLoad, _
                mstrSource, _
                LoadResString(errSetDBBeforeLoad)
    End If

    dtStart = Now

    ' Select based on the global step id since there might
    ' be constraints for a global step that run are executed
    ' for the workspace
    ' This method has the advantage that if the steps are queried right,
everything else follows
    strSql = "Select a.constraint_id, a.step_id, a.version_no, " & _
        " a.constraint_type, a.global_step_id, a.global_version_no, " &
_
        " a.sequence_no, b.workspace_id " & _
        " from step_constraints a, att_steps b " & _
        " where a.global_step_id = b.step_id " & _
        " and a.global_version_no = b.version_no " & _
```

```
        " and a.global_step_id = [g_s_id] " & _
        " and a.global_version_no = [g_ver_no] " & _
        " and b.archived_flag = [archived] "

    ' Find the highest X-component of the version number
    strSql = strSql & " AND ( a.step_id = 0 or ( cint( mid( a.version_no,
1, instr( a.version_no, " & gstrDQ & gstrVerSeparator & gstrDQ & " ) -
1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS d " & _
        " WHERE a.step_id = d.step_id " & _
        " and d.archived_flag = [archived] ) "

    ' Find the highest Y-component of the version number for the highest
X-component
    strSql = strSql & " AND cint( mid( a.version_no, instr( a.version_no,
" & gstrDQ & gstrVerSeparator & gstrDQ & " ) + 1 ) ) = " & _
        " ( select max( cint( mid( version_no, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) + 1 ) ) ) " & _
        " from att_steps AS y " & _
        " Where a.step_id = y.step_id " & _
        " AND cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS c " & _
        " WHERE y.step_id = c.step_id " & _
        " and c.archived_flag = [archived] ) ) ) "

    ' Order the constraints by sequence within a given step
    strSql = strSql & " order by a.sequence_no "

    Set qyCons = mcarrConstraints.NodeDB.CreateQueryDef(gstrEmptyString,
strSql)
    qyCons.Parameters("archived").Value = False

    rstStepsInWsp.MoveFirst

    While Not rstStepsInWsp.EOF

        If Not (rstStepsInWsp!global_flag) Then
            qyCons.Close
            BugMessage "Query constraints Read + load took: " &
CStr(DateDiff("s", dtStart, Now))
            Exit Sub
        End If

        qyCons.Parameters("g_s_id").Value = rstStepsInWsp!step_id
        qyCons.Parameters("g_ver_no").Value = rstStepsInWsp!version_no

        Set recConstraints = qyCons.OpenRecordset(dbOpenSnapshot)

        Call LoadRecordsetInConstraintArray(recConstraints)
        recConstraints.Close

        rstStepsInWsp.MoveNext
    Wend

    qyCons.Close
    BugMessage "Query constraints Read + load took: " &
CStr(DateDiff("s", dtStart, Now))

    Exit Sub

LoadConstraintsErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadConstraints"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadDataFailed, _
            mstrSource, _
            LoadResString(errLoadDataFailed)

End Sub
Public Sub UnloadStepConstraints(ByVal lngStepId As Long)

    ' Unloads all the constraints for the workspace from
    ' the constraints array

    Dim lngIndex As Long

    ' Find all constraints in the array with a matching step id
    ' It is important to step in reverse order through the array,
    ' since we delete constraint records!
    For lngIndex = mcarrConstraints.Count - 1 To 0 Step -1
        If mcarrConstraints(lngIndex).GlobalStepId = lngStepId Then

            ' Unload the constraint from the array
            Call mcarrConstraints.Unload(lngIndex)

        End If
    Next lngIndex

End Sub
Public Sub UnloadConstraint(cOldConstraint As cConstraint)
    ' Unloads the constraint from the constraints array

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryConstraintIndex(cOldConstraint.ConstraintId)

    Call mcarrConstraints.Unload(lngDeleteElement)

End Sub
Private Sub LoadRecordsetInConstraintArray(ByVal recConstraints As
Recordset)
    ' Loads all the constraint records in the passed in
    ' recordset into the array

    Dim cNewConstraint As cConstraint

    On Error GoTo LoadRecordsetInConsArrayErr
```

```
    mstrSource = mstrModuleName & "LoadRecordsetInConstraintArray"

    If recConstraints.RecordCount = 0 Then
        Exit Sub
    End If

    recConstraints.MoveFirst
    While Not recConstraints.EOF
        Set cNewConstraint = New cConstraint

        ' Initialize Constraint values
        cNewConstraint.ConstraintId =
CLng(ErrorOnNullField(recConstraints, "Constraint_id"))
        cNewConstraint.StepId = CLng(ErrorOnNullField(recConstraints,
"step_id"))
        cNewConstraint.VersionNo = CStr(ErrorOnNullField(recConstraints,
"version_no"))

        cNewConstraint.GlobalStepId =
CLng(ErrorOnNullField(recConstraints, "global_step_id"))
        cNewConstraint.GlobalVersionNo =
CStr(ErrorOnNullField(recConstraints, "global_version_no"))
        cNewConstraint.SequenceNo = CInt(ErrorOnNullField(recConstraints,
"sequence_no"))

        cNewConstraint.WorkspaceId =
CLng(ErrorOnNullField(recConstraints, FLD_ID_WORKSPACE))
        cNewConstraint.ConstraintType =
CInt(ErrorOnNullField(recConstraints, "constraint_type"))

        ' Add this record to the array of Constraints
        mcarrConstraints.Load cNewConstraint

        Set cNewConstraint = Nothing
        recConstraints.MoveNext
    Wend

    Exit Sub

LoadRecordsetInConsArrayErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadRecordsetInConstraintArray"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, _
            mstrSource, _
            LoadResString(errLoadRsInArrayFailed)

End Sub

Public Function ConstraintsForStep( _
        ByVal lngStepId As Long, _
        ByVal strVersionNo As String, _
        Optional ByVal intConstraintType As ConstraintType = 0, _
        Optional ByVal blnSort As Boolean = True, _
        Optional ByVal blnGlobal As Boolean = False, _
        Optional ByVal blnGlobalConstraintsOnly As Boolean = False) _
    As Variant

' Returns a variant containing an array of cConstraint objects,
' containing all the constraints that have been defined for the
' given step. If the Global flag is set to true, the
' search will be made for all the constraints that have
' a matching global_step_id

Dim lngIndex As Long
Dim cStepConstraint() As cConstraint
Dim lngConstraintCount As Long
Dim cTempConstraint As cConstraint

On Error GoTo ConstraintsForStepErr
mstrSource = mstrModuleName & "ConstraintsForStep"

lngConstraintCount = 0

' Find each element in the constraints array
For lngIndex = 0 To mcarrConstraints.Count - 1
    ' If a constraint type has been specified then check
    ' if the constraint type for the record matches the
    ' passed in type
    Set cTempConstraint = mcarrConstraints(lngIndex)
    If Not blnGlobal Then
        If cTempConstraint.StepId = lngStepId And _
                cTempConstraint.VersionNo = strVersionNo And _
                cTempConstraint.IndOperation <> DeleteOp And _
                (intConstraintType = 0 Or _
                cTempConstraint.ConstraintType = intConstraintType)
Then
            ' We have a matching constraint for the given step
            AddArrayElement cStepConstraint, _
                    cTempConstraint, lngConstraintCount
        End If
    Else
        If cTempConstraint.GlobalStepId = lngStepId And _
                cTempConstraint.GlobalVersionNo = strVersionNo And _
                cTempConstraint.IndOperation <> DeleteOp Then
            If blnGlobalConstraintsOnly = False Or _
                    (blnGlobalConstraintsOnly And _
                    cTempConstraint.StepId = 0 And _
                    cTempConstraint.VersionNo = gstrMinVersion) Then

                ' We have a matching constraint for the global step
                AddArrayElement cStepConstraint, _
                        cTempConstraint, lngConstraintCount
            End If
        End If
    End If

Next lngIndex

' Set the return value of the function to the array of
' constraints that has been built above
If lngConstraintCount = 0 Then
```

```
        ConstraintsForStep = Empty
    Else
        ConstraintsForStep = cStepConstraint()
    End If

    ' Sort the constraints
    If blnSort Then
        Call QuickSort(ConstraintsForStep)
    End If

    Exit Function

ConstraintsForStepErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errConstraintsForStepFailed, _
            mstrSource, _
            LoadResString(errConstraintsForStepFailed)

End Function
Private Sub AddArrayElement(ByRef arrNodes() As cConstraint, _
        ByVal objToAdd As cConstraint, _
        ByRef lngCount As Long)
    ' Adds the passed in object to the array

    ' Increase the array dimension and add the object to it
    ReDim Preserve arrNodes(lngCount)
    Set arrNodes(lngCount) = objToAdd
    lngCount = lngCount + 1

End Sub

Public Function ConstraintsForWsp( _
        ByVal lngWorkspaceId As Long, _
        Optional ByVal intConstraintType As Integer = 0, _
        Optional ByVal blnSort As Boolean = True, _
        Optional ByVal blnGlobalConstraintsOnly As Boolean = False) _
        As Variant

    ' Returns a variant containing an array of cConstraint objects,
    ' containing all the constraints that have been defined for the
    ' given workspace.

    Dim lngIndex As Long
    Dim cWspConstraint() As cConstraint
    Dim lngConstraintCount As Long
    Dim cTempConstraint As cConstraint

    On Error GoTo ConstraintsForWspErr
    mstrSource = mstrModuleName & "ConstraintsForWsp"

    lngConstraintCount = 0

    ' Find each element in the constraints array
    For lngIndex = 0 To mcarrConstraints.Count - 1
        ' If a constraint type has been specified then check
        ' if the constraint type for the record matches the
        ' passed in type
        Set cTempConstraint = mcarrConstraints(lngIndex)
        If cTempConstraint.WorkspaceId = lngWorkspaceId And _
                cTempConstraint.IndOperation <> DeleteOp And _
                (intConstraintType = 0 Or _
                cTempConstraint.ConstraintType = intConstraintType) Then

            If blnGlobalConstraintsOnly = False Or _
                    (blnGlobalConstraintsOnly And _
                    cTempConstraint.StepId = 0 And _
                    cTempConstraint.VersionNo = gstrMinVersion) Then

                ' We have a matching constraint for the workspace
                AddArrayElement cWspConstraint, _
                        cTempConstraint, lngConstraintCount
            End If
        End If
    Next lngIndex

    ' Set the return value of the function to the array of
    ' constraints that has been built above
    If lngConstraintCount = 0 Then
        ConstraintsForWsp = Empty
    Else
        ConstraintsForWsp = cWspConstraint()
    End If

    ' Sort the constraints
    If blnSort Then
        Call QuickSort(ConstraintsForWsp)
    End If

    Exit Function

ConstraintsForWspErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errConstraintsForWspFailed, _
            mstrSource, _
            LoadResString(errConstraintsForWspFailed)

End Function
Public Function PreConstraintsForStep( _
        ByVal lngStepId As Long, _
        ByVal strVersionNo As String, _
        Optional ByVal blnSort As Boolean) As Variant

    ' Returns a variant containing an array of cConstraint objects,
    ' containing all the pre-execution constraints that have
    ' been defined for the given step_id and version

    ' Call a function that will return a variant containing
    ' all the constraints of the passed in type
    PreConstraintsForStep = ConstraintsForStep(lngStepId, _
            strVersionNo, gintPreStep, blnSort)
```

```
End Function
Public Function PostConstraintsForStep( _
        ByVal lngStepId As Long, _
        ByVal strVersionNo As String, _
        Optional ByVal blnSort As Boolean) As Variant

    ' Returns a variant containing an array of cConstraint objects,
    ' containing all the Post-execution constraints that have
    ' been defined for the given step_id and version

    ' Call a function that will return a variant containing
    ' all the constraints of the passed in type
    PostConstraintsForStep = ConstraintsForStep(lngStepId, _
            strVersionNo, gintPostStep, blnSort)

End Function
Public Function PostConstraintsForWsp( _
        ByVal lngWorkspaceId As Long, _
        Optional ByVal blnSort As Boolean) As Variant

    ' Returns a variant containing an array of cConstraint objects,
    ' containing all the Post-execution globals that have
    ' been defined for the workspace

    ' Call a function that will return a variant containing
    ' all the constraints of the passed in type
    PostConstraintsForWsp = ConstraintsForWsp(lngWorkspaceId, _
            gintPostStep, blnSort, True)

End Function
Public Function PreConstraintsForWsp( _
        ByVal lngWorkspaceId As Long, _
        Optional ByVal blnSort As Boolean) As Variant

    ' Returns a variant containing an array of cConstraint objects,
    ' containing all the Pre-execution globals that have
    ' been defined for the workspace

    ' Call a function that will return a variant containing
    ' all the constraints of the passed in type
    PreConstraintsForWsp = ConstraintsForWsp(lngWorkspaceId, _
            gintPreStep, blnSort, True)

End Function
Public Property Get ConstraintCount() As Long

    ConstraintCount = mcarrConstraints.Count

End Property
```

cArrParameters.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cArrParameters"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cArrParameters.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cParameter objects.
'              Type-safe wrapper around cNodeCollections.
'              Also contains additional functions to determine parameter
'              values, validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrParameters As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cArrParameters."

Public Sub InitBuiltInsForRun(lWspId As Long, lRunId As Long)

    Dim cParamRec As cParameter

    ' Initialize the values of the run_id and output_dir built-in
parameters and save them
    ' to the database
    Set cParamRec = GetParameterValue(lWspId, PARAM_RUN_ID)
    cParamRec.ParameterValue = CStr(lRunId)
    Call Modify(cParamRec)

    Set cParamRec = GetParameterValue(lWspId, PARAM_OUTPUT_DIR)
    cParamRec.ParameterValue = GetDefaultDir(lWspId, Me)
    cParamRec.ParameterValue = cParamRec.ParameterValue &
gstrFileSeparator & CStr(lRunId)
    Call Modify(cParamRec)

    Call SaveParametersInWsp(lWspId)

End Sub


Public Property Set ParamDatabase(vdata As Database)

    Set mcarrParameters.NodeDB = vdata

End Property
Public Sub Modify(cModifiedParam As cParameter)
```

```
    ' First check if the parameter record is valid
    Call CheckDupParamName(cModifiedParam)

    Call mcarrParameters.Modify(cModifiedParam)

End Sub
Public Sub Load(ByRef cParamToAdd As cParameter)

    Call mcarrParameters.Load(cParamToAdd)

End Sub
Public Sub Add(ByRef cParamToAdd As cParameter)

    Set cParamToAdd.NodeDB = mcarrParameters.NodeDB

    ' First check if the parameter record is valid
    Call Validate(cParamToAdd)

    ' Retrieve a unique parameter identifier
    cParamToAdd.ParameterId = cParamToAdd.NextIdentifier

    Call mcarrParameters.Add(cParamToAdd)

End Sub

Public Sub Unload(lngParamToDelete As Long)

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lngParamToDelete)

    Call mcarrParameters.Unload(lngDeleteElement)

End Sub

Public Sub SaveParametersInWsp(ByVal lngWorkspace As Long)
    ' Calls a procedure to commit all changes to the parameters
    ' for the passed in workspace.

    ' Call a procedure to save all parameter records for the
    ' workspace
    Call mcarrParameters.Save(lngWorkspace)

End Sub
Public Function GetParameterValue(ByVal lngWorkspace As Long, _
        ByVal strParamName As String) As cParameter
    ' Returns the value for the passed in workspace parameter

    Dim cParamRec As cParameter
    Dim lngIndex As Long

    On Error GoTo GetParameterValueErr

    ' Find all parameters in the array with a matching workspace id
    For lngIndex = 0 To mcarrParameters.Count - 1
        Set cParamRec = mcarrParameters(lngIndex)
        If cParamRec.WorkspaceId = lngWorkspace And _
                cParamRec.ParameterName = strParamName Then

            Set GetParameterValue = cParamRec
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcarrParameters.Count - 1 Then
        ' The parameter has not been defined for the workspace
        ' Raise an error
        On Error GoTo 0
        Err.Raise vbObjectError + errParamNameInvalid, _
                mstrModuleName & "GetParameterValue", _
                LoadResString(errParamNameInvalid)
    End If

    Exit Function

GetParameterValueErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "GetParameterValue"
    On Error GoTo 0
    Err.Raise vbObjectError + errGetParamValueFailed, _
            gstrSource, _
            LoadResString(errGetParamValueFailed)

End Function
Public Sub Delete(lngParamToDelete As Long)
    ' Delete the passed in parameter

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lngParamToDelete)
    Call mcarrParameters.Delete(lngDeleteElement)

End Sub
Private Function QueryIndex(lngParameterId As Long) As Long

    Dim lngIndex As Long

    ' Find the matching parameter record in the array
    For lngIndex = 0 To mcarrParameters.Count - 1
        If mcarrParameters(lngIndex).ParameterId = lngParameterId And _
                mcarrParameters(lngIndex).IndOperation <> DeleteOp Then
            QueryIndex = lngIndex
            Exit Function
        End If
    Next lngIndex

    ' Raise error that parameter has not been found
    On Error GoTo 0
    Err.Raise vbObjectError + errParamNotFound,
"cArrParameters.QueryIndex", _
        LoadResString(errParamNotFound)
```

```
End Function

Public Function QueryParameter(lngParameterId As Long) _
        As cParameter

    Dim lngQueryElement As Long

    lngQueryElement = QueryIndex(lngParameterId)

    ' Return the queried parameter object
    Set QueryParameter = mcarrParameters(lngQueryElement)

End Function
Public Property Get ParameterCount() As Long

    ParameterCount = mcarrParameters.Count

End Property
Public Property Get Item(lngIndex As Long) As cParameter
Attribute Item.VB_UserMemId = 0

    Set Item = mcarrParameters(lngIndex)

End Property

Public Sub Validate(ByVal cParamToValidate As cParameter)
    ' This procedure is necessary since the class cannot validate
    ' all the parameter properties on it's own. This is 'coz we
    ' might have created new parameters in the workspace, but not
    ' saved them to the database yet - hence the duplicate check
    ' has to be repeated in the array

    Dim lngIndex As Long
    Dim cTempParam As cParameter

    On Error GoTo ValidateErr

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrParameters.Count - 1
        Set cTempParam = mcarrParameters(lngIndex)
        If cTempParam.WorkspaceId = cParamToValidate.WorkspaceId And _
                cTempParam.ParameterName =
cParamToValidate.ParameterName And _
                cTempParam.IndOperation <> DeleteOp Then
            On Error GoTo 0
            Err.Raise vbObjectError + errDuplicateParameterName, _
                mstrSource, LoadResString(errDuplicateParameterName)
        End If
    Next lngIndex

    Exit Sub

ValidateErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Validate"
    On Error GoTo 0
    Err.Raise vbObjectError + errValidateFailed, _
        mstrSource, LoadResString(errValidateFailed)

End Sub
Public Sub CheckDupParamName(ByVal cParamToValidate As cParameter)

    Dim lngIndex As Long
    Dim cTempParam As cParameter

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrParameters.Count - 1
        Set cTempParam = mcarrParameters(lngIndex)
        If cTempParam.WorkspaceId = cParamToValidate.WorkspaceId And _
                cTempParam.ParameterName =
cParamToValidate.ParameterName And _
                cTempParam.ParameterId <> cParamToValidate.ParameterId
And _
                cTempParam.IndOperation <> DeleteOp Then
            ShowError errDuplicateParameterName
            On Error GoTo 0
            Err.Raise vbObjectError + errDuplicateParameterName, _
                mstrSource, LoadResString(errDuplicateParameterName)
        End If
    Next lngIndex

End Sub

Private Sub Class_Initialize()

    'bugmessage "cArrParameters: Initialize event - setting parameter
count to 0"
    Set mcarrParameters = New cNodeCollections

End Sub


Private Sub Class_Terminate()

    Set mcarrParameters = Nothing

End Sub
```

cArrSteps.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cArrSteps"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cArrSteps.cls
'              Microsoft TPC-H Kit Ver. 1.00
```

```
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cStep objects.
'              Type-safe wrapper around cNodeCollections.
'              Also contains additional functions to update parent
version
'              on substeps, validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrSteps As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cArrSteps."

Public Sub Unload(lngStepToDelete As Long)

    Dim lngDeleteElement As Long
    Dim cUnloadStep As cStep

    lngDeleteElement = QueryStepIndex(lngStepToDelete)
    Set cUnloadStep = QueryStep(lngStepToDelete)

    ' First unload all iterators for the step
    Call cUnloadStep.UnloadIterators

    ' Unload the step from the collection
    Call mcarrSteps.Unload(lngDeleteElement)

End Sub
Public Sub Modify(cModifiedStep As cStep)

    Dim iAppend As Integer
    Dim sLabel As String

    On Error GoTo ModifyErr

    iAppend = 0
    sLabel = cModifiedStep.StepLabel

    Validate cModifiedStep

    Call mcarrSteps.Modify(cModifiedStep)

    Exit Sub

ModifyErr:
    ' If the error raised by the add function is due to a duplication
    ' of the step label, then try to generate a unique label
    If Err.Number - vbObjectError = errStepLabelUnique Then
        iAppend = iAppend + 1
        cModifiedStep.StepLabel = sLabel & CStr(iAppend)
        ' Try to insert the step record again
        Resume
    End If

    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyStepFailed, _
            gstrSource, _
            LoadResString(errModifyStepFailed)

End Sub
Public Sub UpdateParentVersion(ByVal lngStepId As Long, _
        ByVal strNewVersion As String, _
        ByVal strOldVersion As String, _
        ByVal intStepType As Integer)

    ' Does all the processing needed to update the parent version
    ' number on all the sub-steps for a given step
    ' It updates the parent version no in the database for all
    ' sub-steps of the passed in step id
    ' It also updates the parent version number on all sub-steps
    ' in the array to the new version passed in

    Dim lngIndex As Long
    Dim cUpdateStep As cStep

    On Error GoTo UpdateParentVersionErr

    If intStepType <> gintManagerStep Then
        ' Only a manager can have sub-steps - if the passed
        ' in step is not a manager, exit
        Exit Sub
    End If

    ' For all steps in the array
    For lngIndex = 0 To mcarrSteps.Count - 1

        Set cUpdateStep = mcarrSteps(lngIndex)

        ' If the current step is a sub-step of the passed in step
        If cUpdateStep.ParentStepId = lngStepId And _
                cUpdateStep.ParentVersionNo = strOldVersion And _
                Not cUpdateStep.ArchivedFlag Then

            ' Update the parent version number for the sub-step
            ' in the array
            cUpdateStep.ParentVersionNo = strNewVersion

            ' Update the parent version number for the sub-step
            ' in the array
            Call Modify(cUpdateStep)

        End If
    Next lngIndex
```

```
    Exit Sub

UpdateParentVersionErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "UpdateParentVersion"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateParentVersionFailed, _
            mstrSource, _
            LoadResString(errUpdateParentVersionFailed)

End Sub
Private Sub Validate(cCheckStep As cStep)

    ' Step validations that depend on other steps in the collection

    Dim lngIndex As Long

    ' Ensure that the step label is unique in the workspace
    For lngIndex = 0 To mcarrSteps.Count - 1

        ' If the current step is a sub-step of the passed in step
        If mcarrSteps(lngIndex).WorkspaceId = cCheckStep.WorkspaceId And
_
                mcarrSteps(lngIndex).StepLabel = cCheckStep.StepLabel
And _
                mcarrSteps(lngIndex).StepId <> cCheckStep.StepId And _
                mcarrSteps(lngIndex).IndOperation <> DeleteOp Then
            On Error GoTo 0
            Err.Raise vbObjectError + errStepLabelUnique, _
                    mstrModuleName & "Validate", _
                    LoadResString(errStepLabelUnique)
        End If
    Next lngIndex

End Sub

Public Sub ValidateStep(cCheckStep As cStep)

    On Error GoTo ValidateStepErr

    'Public wrapper for Validate function (2 many Validates)
    Call Validate(cCheckStep)

    Exit Sub

ValidateStepErr:
    ShowError errStepLabelUnique
    On Error GoTo 0
    Err.Raise vbObjectError + errValidateFailed, _
            gstrSource, _
            LoadResString(errValidateFailed)
End Sub

Private Sub Class_Initialize()

    BugMessage "cArrSteps: Initialize event - setting step count to 0"
    Set mcarrSteps = New cNodeCollections

End Sub

Private Sub Class_Terminate()

    BugMessage "cArrSteps: Terminate event triggered"
    Set mcarrSteps = Nothing

End Sub

Public Sub Add(ByVal cStepToAdd As cStep)

    Dim iAppend As Integer
    Dim sLabel As String

    On Error GoTo AddErr

    iAppend = 0
    sLabel = cStepToAdd.StepLabel

    Set cStepToAdd.NodeDB = mcarrSteps.NodeDB

    ' Retrieve a unique step identifier
    cStepToAdd.StepId = cStepToAdd.NextStepId

    Validate cStepToAdd

    ' Call a procedure to add the step record
    Call mcarrSteps.Add(cStepToAdd)

    ' Call a procedure to add all iterators for the step
    cStepToAdd.AddAllIterators

    Exit Sub

AddErr:
    ' If the error raised by the add function is due to a duplication
    ' of the step label, then try to generate a unique label
    If Err.Number - vbObjectError = errStepLabelUnique Then
        iAppend = iAppend + 1
        cStepToAdd.StepLabel = sLabel & CStr(iAppend)
        ' Try to insert the step record again
        Resume
    End If

    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertStepFailed, _
            gstrSource, _
            LoadResString(errInsertStepFailed)

End Sub
Public Sub Load(cStepToLoad As cStep)
```

```
    Call mcarrSteps.Load(cStepToLoad)

End Sub
Public Sub SaveStepsInWsp(ByVal lngWorkspace As Long)
    ' Calls a procedure to commit all changes to the steps
    ' in the passed in workspace.

    Dim lngIndex As Integer

    ' Find all steps in the array with a matching workspace id
    ' It is important to step in reverse order through the array,
    ' since we delete step records sometimes!
    For lngIndex = mcarrSteps.Count - 1 To 0 Step -1
        If mcarrSteps(lngIndex).WorkspaceId = lngWorkspace Then

            ' Call a procedure to commit all changes to the
            ' Step record, if any
            Call CommitStep(mcarrSteps(lngIndex), lngIndex)

        End If
    Next lngIndex

End Sub
Private Sub CommitStep(ByVal cCommitStep As cStep, _
        ByVal intIndex As Integer)
    ' This procedure checks if any changes have been made to the
    ' passed in Step. If so, it calls the step methods to commit
    ' the changes.

    ' First commit all changes to the iterator records for
    ' the step
    cCommitStep.SaveIterators

    Call mcarrSteps.Commit(cCommitStep, intIndex)

End Sub
Public Sub Delete(lngStepToDelete As Long)

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryStepIndex(lngStepToDelete)
    Call mcarrSteps.Delete(lngDeleteElement)

End Sub
Public Function QueryStepIndex(lngStepId As Long) As Long

    Dim lngIndex As Long

    ' Find the element in the array that corresponds to the
    ' passed in step id - note that while there will be multiple
    ' versions of a step in the database, only one version will
    ' be currently loaded in the array - meaning that the stepid
    ' is enough to uniquely identify a step
    For lngIndex = 0 To mcarrSteps.Count - 1
        If mcarrSteps(lngIndex).StepId = lngStepId Then
            QueryStepIndex = lngIndex
            Exit Function
        End If
    Next lngIndex

    ' Raise error that step has not been found
    On Error GoTo 0
    Err.Raise vbObjectError + errStepNotFound, mstrSource, _
        LoadResString(errStepNotFound)

End Function

Public Function QueryStep(ByVal lngStepId As Long) As cStep

    ' Populates the passed in cStep object with the property
    ' values corresponding to the Step Identifier, lngStepId

    Dim lngQueryElement As Integer

    lngQueryElement = QueryStepIndex(lngStepId)

    ' Initialize the passed in step object to the queried step
    Set QueryStep = mcarrSteps(lngQueryElement)

End Function
Public Property Get Item(ByVal Position As Long) As cStep
Attribute Item.VB_UserMemId = 0

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mcarrSteps.Count Then
        Set Item = mcarrSteps(Position)
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Set Item(ByVal Position As Long, _
        ByVal cStepRec As cStep)

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mcarrSteps.Count Then
        Set mcarrSteps(Position) = cStepRec
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Set StepDB(vdata As Database)

    Set mcarrSteps.NodeDB = vdata

End Property
```

```
Public Function SubSteps(ByVal lngStepId As Long, _
        ByVal strVersionNo As String) As Variant

    ' Returns a variant containing an array of all the substeps
    ' for the passed in step

    Dim intIndex As Integer
    Dim cSubSteps() As cStep
    Dim lngStepCount As Long
    Dim cQueryStep As cStep

    On Error GoTo SubStepsErr

    lngStepCount = 0

    Set cQueryStep = QueryStep(lngStepId)

    ' Only a manager can have sub-steps
    If cQueryStep.StepType = gintManagerStep Then

        ' For each element in the Steps array
        For intIndex = 0 To mcarrSteps.Count - 1
            ' Check if the parent step id and parent version number
            ' match the passed in step
            If mcarrSteps(intIndex).ParentStepId = lngStepId And _
                    mcarrSteps(intIndex).ParentVersionNo = strVersionNo
And _
                    mcarrSteps(intIndex).IndOperation <> DeleteOp Then

                ' Increase the array dimension and add the step
                ' to it
                ReDim Preserve cSubSteps(lngStepCount)
                Set cSubSteps(lngStepCount) = mcarrSteps(intIndex)
                lngStepCount = lngStepCount + 1

            End If
        Next intIndex

    End If

    ' Set the return value of the function to the array of
    ' Steps that has been built above
    If lngStepCount = 0 Then
        SubSteps = Empty
    Else
        SubSteps = cSubSteps()
    End If

    Exit Function

SubStepsErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "SubSteps"
    On Error GoTo 0
    Err.Raise vbObjectError + errSubStepsFailed, _
            mstrSource, _
            LoadResString(errSubStepsFailed)

End Function

Public Property Get StepCount() As Integer

    StepCount = mcarrSteps.Count

End Property
```

cAsyncShell.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cAsyncShell"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'--------------------------------------------------------------------------
--------
' Copyright □ 1997 Microsoft Corporation. All rights reserved.
'
' You have a royalty-free right to use, modify, reproduce and distribute
the
' Sample Application Files (and/or any modified version) in any way you
find
' useful, provided that you agree that Microsoft has no warranty,
obligations or
' liability for any Sample Application Files.
'--------------------------------------------------------------------------
--------

Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cAsyncShell."

Public Event Terminated()

Private WithEvents moTimer As cTimerSM
Attribute moTimer.VB_VarHelpID = -1
Private proc As PROCESS_INFORMATION
Private mfShelling As Boolean


'--------------------------------------------------------------------------
--------
'Initialization and cleanup:

Private Sub Class_Initialize()
    Set moTimer = New cTimerSM
```

```
End Sub

Private Sub Class_Terminate()
    If mfShelling Then CloseHandle proc.hProcess
End Sub

'------------------------------------------------------------------------
--------
'Shelling:

Public Sub Shell(CommandLine As String, Optional PollingInterval As Long
= 1000)
    Dim Start As STARTUPINFO

    If mfShelling Then
        On Error GoTo 0
        Err.Raise vbObjectError + errInstanceInUse, _
                mstrSource, _
                LoadResString(errInstanceInUse)
    End If
    mfShelling = True

    ' Initialize the STARTUPINFO structure:
    Start.cb = Len(Start)
    Start.dwFlags = STARTF_USESHOWWINDOW
    Start.wShowWindow = SW_SHOWMINNOACTIVE

    ' Start the shelled application:
    CreateProcessA 0&, CommandLine, 0&, 0&, 1&, _
        NORMAL_PRIORITY_CLASS, 0&, 0&, Start, proc

    With moTimer
        If PollingInterval > 0 Then
            .Interval = PollingInterval
        Else
            .Interval = 1000
        End If
        .Enabled = True
    End With
End Sub
'------------------------------------------------------------------------
--------
'Aborting:
Public Sub Abort()
    Dim nCode As Long
'    Dim X As Integer
'    Dim ReturnVal As Integer

    On Error GoTo AbortErr

    If Not mfShelling Then
        Call WriteError(errProgramError, mstrSource)
    Else
'        If IsWindow(proc.hProcess) = False Then Exit Sub
'
'        If (GetWindowLong(proc.hProcess, GWL_STYLE) And WS_DISABLED)
Then Exit Sub
'
'        If IsWindow(proc.hProcess) Then
'            If Not (GetWindowLong(proc.hProcess, GWL_STYLE) And
WS_DISABLED) Then
'                X = PostMessage(proc.hProcess, WM_CANCELMODE, 0, 0&)
'                X = PostMessage(proc.hProcess, WM_CLOSE, 0, 0&)
'            End If
'        End If

        If TerminateProcess(proc.hProcess, 0&) = 0 Then
            Debug.Print "Unable to terminate process: " & proc.hProcess
            Call WriteError(errTerminateProcessFailed, mstrSource, _
                    ApiError(GetLastError()))
        Else
            ' Should always come here!
            GetExitCodeProcess proc.hProcess, nCode
            If nCode = STILL_ACTIVE Then
                ' Write an error and close the handles to the
                ' process anyway
                Call WriteError(errTerminateProcessFailed, mstrSource)
            End If
        End If

        ' Close all open handles to the shelled process, even
        ' if any of the above calls error out
        CloseHandle proc.hProcess
        moTimer.Enabled = False
        mfShelling = False
        RaiseEvent Terminated

    End If

    Exit Sub

AbortErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "Abort"
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, _
        mstrSource, _
        LoadResString(errProgramError)

End Sub
Private Sub moTimer_Timer()
    Dim nCode As Long

    GetExitCodeProcess proc.hProcess, nCode
    If nCode <> STILL_ACTIVE Then
        CloseHandle proc.hProcess
        moTimer.Enabled = False
        mfShelling = False
        RaiseEvent Terminated
    End If
End Sub
```

cConnDtl.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConnDtl"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cConnDtl.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a connection.
'               Contains functions to insert, update and delete
'               connection_dtls records from the database.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
Option Base 0

' Local variable(s) to hold property value(s)
Public WorkspaceId As Long
Public ConnNameId As Long
Public ConnName As String
Public ConnectionString As String
Public ConnType As ConnectionType
Public Position As Long
Public NodeDB As Database

Private mintOperation As Operation

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cConnDtl."

' The cSequence class is used to generate unique Connection identifiers
Private mConnectionSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM

Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to differentiate them from the
field names.
    ' When the parameter names are the same as the field names,
parameters in the where
    ' clause do not get created.
    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[w_id]"
                prmParam.Value = WorkspaceId

            Case "[c_id]"
                prmParam.Value = ConnNameId

            Case "[c_name]"
                prmParam.Value = ConnName

            Case "[c_str]"
                prmParam.Value = ConnectionString

            Case "[c_type]"
                prmParam.Value = ConnType

            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource,
prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrModuleName &
"AssignParameters", _
                          LoadResString(errInvalidParameter)
        End Select
    Next prmParam

    Exit Sub

AssignParametersErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrModuleName & "AssignParameters",
LoadResString(errAssignParametersFailed)

End Sub

Public Function Clone() As cConnDtl

    ' Creates a copy of a given Connection

    Dim cCloneConn As cConnDtl

    On Error GoTo CloneErr

    Set cCloneConn = New cConnDtl

    ' Copy all the Connection properties to the newly created Connection
    cCloneConn.WorkspaceId = WorkspaceId
```

```
    cCloneConn.ConnNameId = ConnNameId
    cCloneConn.ConnName = ConnName
    cCloneConn.ConnectionString = ConnectionString
    cCloneConn.ConnType = ConnType
    cCloneConn.IndOperation = mintOperation
    cCloneConn.Position = Position

    ' And set the return value to the newly created Connection
    Set Clone = cCloneConn
    Set cCloneConn = Nothing

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, mstrSource,
LoadResString(errCloneFailed)

End Function
Private Sub CheckDupConnectionName()
    ' Check if the Connection name already exists in the workspace

    Dim rstConnection As Recordset
    Dim strSql As String
    Dim qy As DAO.QueryDef

    On Error GoTo CheckDupConnectionNameErr
    mstrSource = mstrModuleName & "CheckDupConnectionName"

    ' Create a recordset object to retrieve the count of all Connections
    ' for the workspace with the same name
    strSql = "Select count(*) as Connection_count " & _
        " from " & TBL_CONNECTION_DTLS & _
        " where " & FLD_ID_WORKSPACE & " = [w_id]" & _
        " and " & FLD_CONN_DTL_CONNECTION_NAME & " = [c_name]" & _
        " and " & FLD_ID_CONN_NAME & " <> [c_id]"

    Set qy = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)
    Call AssignParameters(qy)

    Set rstConnection = qy.OpenRecordset(dbOpenForwardOnly)

    If rstConnection![Connection_count] > 0 Then
        rstConnection.Close
        qy.Close
        ShowError errDupConnDtlName
        On Error GoTo 0
        Err.Raise vbObjectError + errDupConnDtlName, _
            mstrSource, LoadResString(errDupConnDtlName)
    End If

    rstConnection.Close
    qy.Close

    Exit Sub

CheckDupConnectionNameErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "CheckDupConnectionName"
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, _
        mstrSource, LoadResString(errProgramError)

End Sub
Public Property Let IndOperation(ByVal vdata As Operation)

    ' The valid operations are define in the cOperations
    ' class. Check if the operation is valid
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            BugAssert True
    End Select

End Property
Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    If ConnName = gstrEmptyString Then

        ShowError errConnectionNameMandatory
        On Error GoTo 0
        ' Propogate this error back to the caller
        Err.Raise vbObjectError + errConnectionNameMandatory, _
            mstrSource, LoadResString(errConnectionNameMandatory)
    End If

    ' Raise an error if the Connection name already exists in the
workspace
    Call CheckDupConnectionName

End Sub
Public Sub Add()

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddErr

    ' Validate the record before trying to insert the record
    Call Validate

    ' Create a temporary querydef object
    strInsert = "insert into " & TBL_CONNECTION_DTLS & _
```

```
            "( " & FLD_ID_WORKSPACE & _
            ", " & FLD_ID_CONN_NAME & _
            ", " & FLD_CONN_DTL_CONNECTION_NAME & _
            ", " & FLD_CONN_DTL_CONNECTION_STRING & _
            ", " & FLD_CONN_DTL_CONNECTION_TYPE & " ) " & _
           " values ( [w_id], [c_id], " & _
           " [c_name], [c_str], [c_type] ) "

    Set qy = dbsAttTool.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to assign the Connection values
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close

    Exit Sub

AddErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertFailed, _
        mstrModuleName & "Add", LoadResString(errInsertFailed)

End Sub
Public Sub Delete()

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr

    strDelete = "delete from " & TBL_CONNECTION_DTLS & _
           " where " & FLD_ID_CONN_NAME & " = [c_id]"
    Set qy = dbsAttTool.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteFailed, _
           mstrModuleName & "Delete", LoadResString(errDeleteFailed)

End Sub

Public Sub Modify()

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo ModifyErr

    ' Validate the updated values before trying to modify the db
    Call Validate

    ' Create a temporary querydef object with the modify string
    strUpdate = "update " & TBL_CONNECTION_DTLS & _
           " set " & FLD_ID_WORKSPACE & " = [w_id], " & _
           FLD_CONN_DTL_CONNECTION_NAME & " = [c_name], " & _
           FLD_CONN_DTL_CONNECTION_STRING & " = [c_str], " & _
           FLD_CONN_DTL_CONNECTION_TYPE & " = [c_type] " & _
           " where " & FLD_ID_CONN_NAME & " = [c_id]"
    Set qy = dbsAttTool.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the Connection values to the
    ' querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

ModifyErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyFailed, _
        mstrModuleName & "Modify", LoadResString(errModifyFailed)

End Sub
Public Property Get NextIdentifier() As Long

    Dim lngNextId As Long

    On Error GoTo NextIdentifierErr

    ' Retrieve the next identifier using the sequence class
    Set mConnectionSeq = New cSequence
    Set mConnectionSeq.IdDatabase = dbsAttTool
    mConnectionSeq.IdentifierColumn = FLD_ID_CONN_NAME
    lngNextId = mConnectionSeq.Identifier
    Set mConnectionSeq = Nothing

    NextIdentifier = lngNextId
    Exit Property

NextIdentifierErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errIdGetFailed, _
        mstrModuleName & "NextIdentifier", LoadResString(errIdGetFailed)

End Property
Public Property Get IndOperation() As Operation
```

```
    IndOperation = mintOperation

End Property

Private Sub Class_Initialize()

    Set mFieldValue = New cStringSM

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

    ConnType = giDefaultConnType

End Sub

Private Sub Class_Terminate()

    Set mFieldValue = Nothing

End Sub
```

cConnDtls.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConnDtls"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cConnDtls.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cConnDtl objects.
'              Type-safe wrapper around cNodeCollections.
'              Also contains additional functions to determine the
connection
'              string value, validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrConnDtls As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cConnDtls."

Public Property Set ConnDb(vdata As Database)

    Set mcarrConnDtls.NodeDB = vdata

End Property
Public Sub Modify(cModifiedConn As cConnDtl)

    ' First check if the parameter record is valid
    Call CheckDupConnName(cModifiedConn)

    Call mcarrConnDtls.Modify(cModifiedConn)

End Sub
Public Sub Load(ByRef cConnToAdd As cConnDtl)

    Call mcarrConnDtls.Load(cConnToAdd)

End Sub
Public Sub Add(ByRef cConnToAdd As cConnDtl)

    ' First check if the record is valid
    Call Validate(cConnToAdd)

    ' Retrieve a unique identifier
    cConnToAdd.ConnNameId = cConnToAdd.NextIdentifier

    Call mcarrConnDtls.Add(cConnToAdd)

End Sub

Public Sub Unload(lConnNameId As Long)

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lConnNameId)

    Call mcarrConnDtls.Unload(lngDeleteElement)

End Sub

Public Sub SaveConnDtlsInWsp(ByVal lngWorkspace As Long)
    ' Call a procedure to save all connection details records for the
workspace
    Call mcarrConnDtls.Save(lngWorkspace)

End Sub
Public Function GetConnectionDtl(ByVal lngWorkspace As Long, _
        ByVal strConnectionName As String) As cConnDtl
    ' Returns the connection dtl for the passed in connection name

    Dim lngIndex As Long

    ' Find all parameters in the array with a matching workspace id
    For lngIndex = 0 To mcarrConnDtls.Count - 1
        If mcarrConnDtls(lngIndex).WorkspaceId = lngWorkspace And _
```

```
                mcarrConnDtls(lngIndex).ConnName = strConnectionName
Then

            Set GetConnectionDtl = mcarrConnDtls(lngIndex)
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcarrConnDtls.Count - 1 Then
        ' The parameter has not been defined for the workspace
        ' Raise an error
        On Error GoTo 0
        Err.Raise vbObjectError + errConnNameInvalid, mstrModuleName &
"GetConnection", _
                LoadResString(errConnNameInvalid)
    End If

End Function
Public Sub Delete(lConnNameId As Long)
    ' Delete the passed in parameter

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lConnNameId)
    Call mcarrConnDtls.Delete(lngDeleteElement)

End Sub
Private Function QueryIndex(lConnNameId As Long) As Long

    Dim lngIndex As Long

    ' Find the matching parameter record in the array
    For lngIndex = 0 To mcarrConnDtls.Count - 1
        If mcarrConnDtls(lngIndex).ConnNameId = lConnNameId And _
                mcarrConnDtls(lngIndex).IndOperation <> DeleteOp Then
            QueryIndex = lngIndex
            Exit Function
        End If
    Next lngIndex

    ' Raise error that parameter has not been found
    On Error GoTo 0
    Err.Raise vbObjectError + errQueryIndexFailed,
"cArrParameters.QueryIndex", _
        LoadResString(errQueryIndexFailed)

End Function

Public Function QueryConnDtl(lConnNameId As Long) As cConnDtl

    Dim lngQueryElement As Long

    lngQueryElement = QueryIndex(lConnNameId)

    ' Return the queried connection object
    Set QueryConnDtl = mcarrConnDtls(lngQueryElement)
End Function
Public Property Get Count() As Long

    Count = mcarrConnDtls.Count

End Property
Public Property Get Item(lngIndex As Long) As cConnDtl
Attribute Item.VB_UserMemId = 0

    Set Item = mcarrConnDtls(lngIndex)

End Property

Private Sub Validate(ByVal cConnToValidate As cConnDtl)
    ' This procedure is necessary since the class cannot validate
    ' all the connection_dtl properties on it's own. This is 'coz we
    ' might have created new connections in the workspace, but not
    ' saved them to the database yet - hence the duplicate check
    ' has to be repeated in the array

    Dim lngIndex As Long
    Dim cTempParam As cConnDtl

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrConnDtls.Count - 1
        Set cTempParam = mcarrConnDtls(lngIndex)
        If cTempParam.WorkspaceId = cConnToValidate.WorkspaceId And _
                cTempParam.ConnName = cConnToValidate.ConnName And _
                cTempParam.IndOperation <> DeleteOp Then
            On Error GoTo 0
            Err.Raise vbObjectError + errDupConnDtlName, _
                mstrSource, LoadResString(errDupConnDtlName)
        End If
    Next lngIndex

End Sub
Private Sub CheckDupConnName(ByVal cConnToValidate As cConnDtl)

    Dim lngIndex As Long
    Dim cTempParam As cConnDtl

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrConnDtls.Count - 1
        Set cTempParam = mcarrConnDtls(lngIndex)
        If cTempParam.WorkspaceId = cConnToValidate.WorkspaceId And _
                cTempParam.ConnName = cConnToValidate.ConnName And _
                cTempParam.ConnNameId <> cConnToValidate.ConnNameId And
_
                cTempParam.IndOperation <> DeleteOp Then
            ShowError errDupConnDtlName
            On Error GoTo 0
            Err.Raise vbObjectError + errDupConnDtlName, _
                mstrSource, LoadResString(errDupConnDtlName)
        End If
    Next lngIndex
```

```
End Sub

Private Sub Class_Initialize()

    Set mcarrConnDtls = New cNodeCollections

End Sub

Private Sub Class_Terminate()

    Set mcarrConnDtls = Nothing

End Sub
```

cConnection.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConnection"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cConnection.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a connection
string.
'               Contains functions to insert, update and delete
'               workspace_connections records from the database.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
Option Base 0

' Local variable(s) to hold property value(s)
Public WorkspaceId As Long
Public ConnectionId As Long
Public ConnectionValue As String
Public Description As String
Public NodeDB As Database
Public Position As Long
Public NoCountDisplay As Boolean
Public NoExecute As Boolean
Public ParseQueryOnly As Boolean
Public QuotedIdentifiers As Boolean
Public AnsiNulls As Boolean
Public ShowQueryPlan As Boolean
Public ShowStatsTime As Boolean
Public ShowStatsIO As Boolean
Public ParseOdbcMsg As Boolean
Public RowCount As Long
Public TsqlBatchSeparator As String
Public QueryTimeOut As Long
Public ServerLanguage As String
Public CharacterTranslation As Boolean
Public RegionalSettings As Boolean

Private mstrConnectionName As String
Private mintOperation As Operation

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cConnection."

' The cSequence class is used to generate unique Connection identifiers
Private mConnectionSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM

Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to differentiate them from the
field names.
    ' When the parameter names are the same as the field names,
parameters in the where
    ' clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[w_id]"
                prmParam.Value = WorkspaceId

            Case "[c_id]"
                prmParam.Value = ConnectionId

            Case "[c_name]"
                prmParam.Value = mstrConnectionName

            Case "[c_value]"
                prmParam.Value = ConnectionValue

            Case "[desc]"
                prmParam.Value = Description

            Case "[no_count]"
                prmParam.Value = NoCountDisplay

            Case "[no_exec]"
                prmParam.Value = NoExecute
```

```
            Case "[parse_only]"
                prmParam.Value = ParseQueryOnly

            Case "[quoted_id]"
                prmParam.Value = QuotedIdentifiers

            Case "[a_nulls]"
                prmParam.Value = AnsiNulls

            Case "[show_qp]"
                prmParam.Value = ShowQueryPlan

            Case "[stats_tm]"
                prmParam.Value = ShowStatsTime

            Case "[stats_io]"
                prmParam.Value = ShowStatsIO

            Case "[parse_odbc]"
                prmParam.Value = ParseOdbcMsg

            Case "[row_cnt]"
                prmParam.Value = RowCount

            Case "[batch_sep]"
                prmParam.Value = TsqlBatchSeparator

            Case "[qry_tmout]"
                prmParam.Value = QueryTimeOut

            Case "[lang]"
                prmParam.Value = ServerLanguage

            Case "[char_trans]"
                prmParam.Value = CharacterTranslation

            Case "[reg_settings]"
                prmParam.Value = RegionalSettings

            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource,
prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrModuleName &
"AssignParameters", _
                        LoadResString(errInvalidParameter)
        End Select
    Next prmParam

    Exit Sub

AssignParametersErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrModuleName & "AssignParameters",
LoadResString(errAssignParametersFailed)

End Sub

Public Function Clone() As cConnection

    ' Creates a copy of a given Connection

    Dim cCloneConn As cConnection

    On Error GoTo CloneErr

    Set cCloneConn = New cConnection

    ' Copy all the Connection properties to the newly
    ' created Connection
    Set cCloneConn.NodeDB = NodeDB
    cCloneConn.WorkspaceId = WorkspaceId
    cCloneConn.ConnectionId = ConnectionId
    cCloneConn.ConnectionName = mstrConnectionName
    cCloneConn.ConnectionValue = ConnectionValue
    cCloneConn.Description = Description
    cCloneConn.IndOperation = mintOperation
    cCloneConn.Position = Position
    cCloneConn.NoCountDisplay = NoCountDisplay
    cCloneConn.NoExecute = NoExecute
    cCloneConn.ParseQueryOnly = ParseQueryOnly
    cCloneConn.QuotedIdentifiers = QuotedIdentifiers
    cCloneConn.AnsiNulls = AnsiNulls
    cCloneConn.ShowQueryPlan = ShowQueryPlan
    cCloneConn.ShowStatsTime = ShowStatsTime
    cCloneConn.ShowStatsIO = ShowStatsIO
    cCloneConn.ParseOdbcMsg = ParseOdbcMsg
    cCloneConn.RowCount = RowCount
    cCloneConn.TsqlBatchSeparator = TsqlBatchSeparator
    cCloneConn.QueryTimeOut = QueryTimeOut
    cCloneConn.ServerLanguage = ServerLanguage
    cCloneConn.CharacterTranslation = CharacterTranslation
    cCloneConn.RegionalSettings = RegionalSettings

    ' And set the return value to the newly created Connection
    Set Clone = cCloneConn
    Set cCloneConn = Nothing

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, mstrSource,
LoadResString(errCloneFailed)
```

```
End Function
Private Sub CheckDupConnectionName()
    ' Check if the Connection name already exists in the workspace

    Dim rstConnection As Recordset
    Dim strSql As String
    Dim qy As DAO.QueryDef

    On Error GoTo CheckDupConnectionNameErr
    mstrSource = mstrModuleName & "CheckDupConnectionName"

    ' Create a recordset object to retrieve the count of all Connections
    ' for the workspace with the same name
    strSql = "Select count(*) as Connection_count " & _
        " from workspace_connections " & _
        " where workspace_id = [w_id]" & _
        " and connection_name = [c_name]" & _
        " and connection_id <> [c_id]"

    Set qy = NodeDB.CreateQueryDef(gstrEmptyString, strSql)
    Call AssignParameters(qy)

    Set rstConnection = qy.OpenRecordset(dbOpenForwardOnly)

    If rstConnection![Connection_count] > 0 Then
        rstConnection.Close
        qy.Close
        ShowError errDuplicateConnectionName
        On Error GoTo 0
        Err.Raise vbObjectError + errDuplicateConnectionName, _
            mstrSource, LoadResString(errDuplicateConnectionName)
    End If

    rstConnection.Close
    qy.Close

    Exit Sub

CheckDupConnectionNameErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "CheckDupConnectionName"
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, _
        mstrSource, LoadResString(errProgramError)

End Sub
Private Sub CheckDB()
    ' Check if the database object has been initialized

    If NodeDB Is Nothing Then
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidDB, _
            mstrModuleName & "CheckDB", LoadResString(errInvalidDB)
    End If

End Sub
Public Property Let ConnectionName(vdata As String)

    If vdata = gstrEmptyString Then

        ShowError errConnectionNameMandatory
        On Error GoTo 0
        ' Propogate this error back to the caller
        Err.Raise vbObjectError + errConnectionNameMandatory, _
            mstrSource, LoadResString(errConnectionNameMandatory)
    Else
        mstrConnectionName = vdata
    End If

End Property

Public Property Let IndOperation(ByVal vdata As Operation)

    ' The valid operations are define in the cOperations
    ' class. Check if the operation is valid
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            BugAssert True
    End Select

End Property
Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    ' Check if the db object is valid
    Call CheckDB

    ' Raise an error if the Connection name already exists in the
workspace
    Call CheckDupConnectionName

End Sub
Public Sub Add()

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddErr

    ' Validate the record before trying to insert the record
    Call Validate

    ' Create a temporary querydef object
    strInsert = "insert into workspace_connections " & _
             "( workspace_id, connection_id, " & _
```

```
            "connection_name, connection_value, " & _
            "description, no_count_display, " & _
            "no_execute, parse_query_only, " & _
            "ANSI_quoted_identifiers, ANSI_nulls, " & _
            "show_query_plan, show_stats_time, " & _
            "show_stats_io, parse_odbc_msg_prefixes, " & _
            "row_count, tsql_batch_separator, " & _
            "query_time_out, server_language, " & _
            "character_translation, regional_settings ) " & _
           " values ( [w_id], [c_id], [c_name], [c_value], " & _
           " [desc], [no_count], [no_exec], [parse_only], " & _
           " [quoted_id], [a_nulls], [show_qp], [stats_tm], " & _
           " [stats_io], [parse_odbc], [row_cnt], [batch_sep], " & _
           " [qry_tmout], [lang], [char_trans], [reg_settings] ) "

    Set qy = NodeDB.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to assign the Connection values
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close

    Exit Sub

AddErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertFailed, _
        mstrModuleName & "Add", LoadResString(errInsertFailed)

End Sub
Public Sub Delete()

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr

    ' Check if the db object is valid
    Call CheckDB

    strDelete = "delete from workspace_connections " & _
            " where connection_id = [c_id]"
    Set qy = NodeDB.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteFailed, _
            mstrModuleName & "Delete", LoadResString(errDeleteFailed)

End Sub

Public Sub Modify()

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo ModifyErr

    ' Validate the updated values before trying to modify the db
    Call Validate

    ' Create a temporary querydef object with the modify string
    strUpdate = "update workspace_connections " & _
            " set workspace_id = [w_id], " & _
            "connection_name = [c_name], " & _
            "connection_value = [c_value], " & _
            "description = [desc], " & _
            "no_count_display = [no_count], " & _
            "no_execute = [no_exec], " & _
            "parse_query_only = [parse_only], " & _
            "ANSI_quoted_identifiers = [quoted_id], " & _
            "ANSI_nulls = [a_nulls], " & _
            "show_query_plan = [show_qp], " & _
            "show_stats_time = [stats_tm], " & _
            "show_stats_io = [stats_io], " & _
            "parse_odbc_msg_prefixes = [parse_odbc], " & _
            "row_count = [row_cnt], " & _
            "tsql_batch_separator = [batch_sep], " & _
            "query_time_out = [qry_tmout], " & _
            "server_language = [lang], " & _
            "character_translation = [char_trans], " & _
            "regional_settings = [reg_settings] " & _
            " where connection_id = [c_id]"
    Set qy = NodeDB.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the Connection values to the
    ' querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

ModifyErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyFailed, _
        mstrModuleName & "Modify", LoadResString(errModifyFailed)
```

```
End Sub
Public Property Get ConnectionName() As String

    ConnectionName = mstrConnectionName

End Property

Public Property Get NextIdentifier() As Long

    Dim lngNextId As Long

    On Error GoTo NextIdentifierErr

    ' First check if the database object is valid
    Call CheckDB

    ' Retrieve the next identifier using the sequence class
    Set mConnectionSeq = New cSequence
    Set mConnectionSeq.IdDatabase = NodeDB
    mConnectionSeq.IdentifierColumn = "connection_id"
    lngNextId = mConnectionSeq.Identifier
    Set mConnectionSeq = Nothing

    NextIdentifier = lngNextId
    Exit Property

NextIdentifierErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errIdGetFailed, _
        mstrModuleName & "NextIdentifier", LoadResString(errIdGetFailed)

End Property
Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property

Private Sub Class_Initialize()

    Set mFieldValue = New cStringSM

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

    ' Initialize connection properties to their default values
    NoCountDisplay = DEF_NO_COUNT_DISPLAY
    NoExecute = DEF_NO_EXECUTE
    ParseQueryOnly = DEF_PARSE_QUERY_ONLY
    QuotedIdentifiers = DEF_ANSI_QUOTED_IDENTIFIERS
    AnsiNulls = DEF_ANSI_NULLS
    ShowQueryPlan = DEF_SHOW_QUERY_PLAN
    ShowStatsTime = DEF_SHOW_STATS_TIME
    ShowStatsIO = DEF_SHOW_STATS_IO
    ParseOdbcMsg = DEF_PARSE_ODBC_MSG_PREFIXES
    RowCount = DEF_ROW_COUNT
    TsqlBatchSeparator = DEF_TSQL_BATCH_SEPARATOR
    QueryTimeOut = DEF_QUERY_TIME_OUT
    ServerLanguage = DEF_SERVER_LANGUAGE
    CharacterTranslation = DEF_CHARACTER_TRANSLATION
    RegionalSettings = DEF_REGIONAL_SETTINGS

End Sub

Private Sub Class_Terminate()

    Set NodeDB = Nothing
    Set mFieldValue = Nothing

End Sub
```

cConnections.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConnections"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cConnections.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cConnection objects.
'              Type-safe wrapper around cNodeCollections.
'              Also contains validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrConnections As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cConnections."

Public Property Set ConnDb(vdata As Database)

    Set mcarrConnections.NodeDB = vdata

End Property
Public Sub Modify(cModifiedConn As cConnection)
```

```
    ' First check if the parameter record is valid
    Call CheckDupConnName(cModifiedConn)

    Call mcarrConnections.Modify(cModifiedConn)

End Sub
Public Sub Load(ByRef cConnToAdd As cConnection)

    Call mcarrConnections.Load(cConnToAdd)

End Sub
Public Sub Add(ByRef cConnToAdd As cConnection)

    Set cConnToAdd.NodeDB = mcarrConnections.NodeDB

    ' First check if the record is valid
    Call Validate(cConnToAdd)

    ' Retrieve a unique identifier
    cConnToAdd.ConnectionId = cConnToAdd.NextIdentifier

    Call mcarrConnections.Add(cConnToAdd)

End Sub

Public Sub Unload(lngConnId As Long)

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lngConnId)

    Call mcarrConnections.Unload(lngDeleteElement)

End Sub

Public Sub SaveConnectionsInWsp(ByVal lngWorkspace As Long)
    ' Call a procedure to save all connection records for the workspace
    Call mcarrConnections.Save(lngWorkspace)

End Sub
Public Function GetConnection(ByVal lngWorkspace As Long, _
        ByVal strConnectionName As String) As cConnection
    ' Returns the connection string for the passed in connection name

    Dim lngIndex As Long

    ' Find all parameters in the array with a matching workspace id
    For lngIndex = 0 To mcarrConnections.Count - 1
        If mcarrConnections(lngIndex).WorkspaceId = lngWorkspace And _
                mcarrConnections(lngIndex).ConnectionName =
strConnectionName Then

            Set GetConnection = mcarrConnections(lngIndex)
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcarrConnections.Count - 1 Then
        ' The parameter has not been defined for the workspace
        ' Raise an error
        On Error GoTo 0
        Err.Raise vbObjectError + errConnNameInvalid, mstrModuleName &
"GetConnection", _
                LoadResString(errConnNameInvalid)
    End If

End Function
Public Sub Delete(lngConnId As Long)
    ' Delete the passed in parameter

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lngConnId)
    Call mcarrConnections.Delete(lngDeleteElement)

End Sub
Private Function QueryIndex(lngConnId As Long) As Long

    Dim lngIndex As Long

    ' Find the matching parameter record in the array
    For lngIndex = 0 To mcarrConnections.Count - 1
        If mcarrConnections(lngIndex).ConnectionId = lngConnId And _
                mcarrConnections(lngIndex).IndOperation <> DeleteOp Then
            QueryIndex = lngIndex
            Exit Function
        End If
    Next lngIndex

    ' Raise error that parameter has not been found
    On Error GoTo 0
    Err.Raise vbObjectError + errQueryIndexFailed,
"cArrParameters.QueryIndex", _
        LoadResString(errQueryIndexFailed)

End Function

Public Function QueryConnection(lngConnId As Long) As cConnection

    Dim lngQueryElement As Long

    lngQueryElement = QueryIndex(lngConnId)

    ' Return the queried connection object
    Set QueryConnection = mcarrConnections(lngQueryElement)

End Function
Public Property Get Count() As Long

    Count = mcarrConnections.Count

End Property
```

```
Public Property Get Item(lngIndex As Long) As cConnection
Attribute Item.VB_UserMemId = 0

    Set Item = mcarrConnections(lngIndex)

End Property

Public Sub Validate(ByVal cConnToValidate As cConnection)
    ' This procedure is necessary since the class cannot validate
    ' all the parameter properties on it's own. This is 'coz we
    ' might have created new parameters in the workspace, but not
    ' saved them to the database yet - hence the duplicate check
    ' has to be repeated in the array

    Dim lngIndex As Long
    Dim cTempParam As cConnection

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrConnections.Count - 1
        Set cTempParam = mcarrConnections(lngIndex)
        If cTempParam.WorkspaceId = cConnToValidate.WorkspaceId And _
                cTempParam.ConnectionName =
cConnToValidate.ConnectionName And _
                cTempParam.IndOperation <> DeleteOp Then
            On Error GoTo 0
            Err.Raise vbObjectError + errDuplicateConnectionName, _
                mstrSource, LoadResString(errDuplicateConnectionName)
        End If
    Next lngIndex

End Sub
Public Sub CheckDupConnName(ByVal cConnToValidate As cConnection)

    Dim lngIndex As Long
    Dim cTempParam As cConnection

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrConnections.Count - 1
        Set cTempParam = mcarrConnections(lngIndex)
        If cTempParam.WorkspaceId = cConnToValidate.WorkspaceId And _
                cTempParam.ConnectionName =
cConnToValidate.ConnectionName And _
                cTempParam.ConnectionId <> cConnToValidate.ConnectionId
And _
                cTempParam.IndOperation <> DeleteOp Then
            ShowError errDuplicateConnectionName
            On Error GoTo 0
            Err.Raise vbObjectError + errDuplicateConnectionName, _
                mstrSource, LoadResString(errDuplicateConnectionName)
        End If
    Next lngIndex

End Sub

Private Sub Class_Initialize()

    Set mcarrConnections = New cNodeCollections

End Sub

Private Sub Class_Terminate()

    Set mcarrConnections = Nothing

End Sub
```

cConstraint.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConstraint"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cConstraint.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a constraint.
'               Contains functions to insert, update and delete
'               step_constraints records from the database.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Module level variables to store the property values
Private mlngConstraintId As Long
Private mlngStepId As Long
Private mstrVersionNo As String
Private mintConstraintType As Integer
Private mlngGlobalStepId As Long
Private mstrGlobalVersionNo As String
Private mintSequenceNo As Integer
Private mdbsConstraintDB As Database
Private mlngWorkspaceId As Integer
Private mintOperation As Operation
Private mlngPosition As Long

' The cSequence class is used to generate unique step identifiers
Private mConstraintSeq As cSequence

Private Const mstrModuleName As String = ".cConstraint."
Private mstrSource As String

Public Enum ConstraintType
    gintPreStep = 1
    gintPostStep = 2
```

```
End Enum

Private Const mstrSQ As String = "'"
Public Property Get WorkspaceId() As Long
    WorkspaceId = mlngWorkspaceId
End Property
Public Property Let WorkspaceId(ByVal vdata As Long)
    mlngWorkspaceId = vdata
End Property

Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property
Public Property Let IndOperation(ByVal vdata As Operation)

    On Error GoTo IndOperationErr
    mstrSource = mstrModuleName & "IndOperation"

    ' The valid operations are define in the cOperations
    ' class. Check if the operation is valid
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errInvalidOperation, _
                    mstrSource, LoadResString(errInvalidOperation)
    End Select

    Exit Property

IndOperationErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "IndOperation"
    On Error GoTo 0
    Err.Raise vbObjectError + errLetOperationFailed, _
        mstrSource, LoadResString(errLetOperationFailed)

End Property

Public Function Clone() As cConstraint

    ' Creates a copy of a given constraint

    Dim cConsClone As cConstraint

    On Error GoTo CloneErr
    mstrSource = mstrModuleName & "Clone"

    Set cConsClone = New cConstraint

    ' Copy all the workspace properties to the newly
    ' created workspace
    cConsClone.ConstraintId = mlngConstraintId
    cConsClone.StepId = mlngStepId
    cConsClone.VersionNo = mstrVersionNo
    cConsClone.ConstraintType = mintConstraintType
    cConsClone.GlobalStepId = mlngGlobalStepId
    cConsClone.GlobalVersionNo = mstrGlobalVersionNo
    cConsClone.SequenceNo = mintSequenceNo
    cConsClone.WorkspaceId = mlngWorkspaceId
    cConsClone.IndOperation = mintOperation

    ' And set the return value to the newly created constraint
    Set Clone = cConsClone

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, _
            mstrSource, LoadResString(errCloneFailed)

End Function

Public Property Get SequenceNo() As Integer

    SequenceNo = mintSequenceNo

End Property

Public Property Let SequenceNo(ByVal vdata As Integer)
    mintSequenceNo = vdata
End Property

Public Sub Add()
    ' Inserts a new step constraint into the database

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddErr

    ' First check if the database object is valid
    Call CheckDB

    ' Any record validations
    Call Validate

    ' Create a temporary querydef object
    strInsert = "insert into step_constraints " & _
            "( constraint_id, step_id, version_no, " & _
            " constraint_type, global_step_id, global_version_no,
sequence_no ) " & _
            " values ( [cons_id], [s_id], [ver_no], " & _
            " [cons_type], [g_step_id], [g_ver_no], " & _
            " [seq_no] )"
```

```
    Set qy = mdbsConstraintDB.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close

'    strInsert = "insert into step_constraints " & _
'        "( constraint_id, step_id, version_no, " & _
'        " constraint_type, global_step_id, global_version_no,
sequence_no ) " & _
'        " values ( " & _
'        Str(mlngConstraintId) & ", " & Str(mlngStepId) & ", " & _
'        mstrSQ & mstrVersionNo & mstrSQ & ", " &
Str(mintConstraintType) & ", " & _
'        Str(mlngGlobalStepId) & ", " & mstrSQ & mstrGlobalVersionNo &
mstrSQ & ", " & _
'        Str(mintSequenceNo) & " ) "
'
'    BugMessage strInsert
'    mdbsConstraintDB.Execute strInsert, dbFailOnError
    Exit Sub

AddErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Add"
    On Error GoTo 0
    Err.Raise vbObjectError + errAddConstraintFailed, _
            mstrSource, _
            LoadResString(errAddConstraintFailed)
End Sub
Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the field names. When the parameter names are
    ' the same as the field names, parameters in the where
    ' clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr
    mstrSource = mstrModuleName & "AssignParameters"

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[cons_id]"
                prmParam.Value = mlngConstraintId

            Case "[s_id]"
                prmParam.Value = mlngStepId

            Case "[ver_no]"
                prmParam.Value = mstrVersionNo

            Case "[cons_type]"
                prmParam.Value = mintConstraintType

            Case "[g_step_id]"
                prmParam.Value = mlngGlobalStepId

            Case "[g_ver_no]"
                prmParam.Value = mstrGlobalVersionNo

            Case "[seq_no]"
                prmParam.Value = mintSequenceNo

            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource, _
                        prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrSource, _
                        LoadResString(errInvalidParameter)
        End Select
    Next prmParam

    Exit Sub

AssignParametersErr:

    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrSource, LoadResString(errAssignParametersFailed)

End Sub
Public Property Get NextIdentifier() As Long

    Dim lngNextId As Long

    On Error GoTo NextIdentifierErr

    ' First check if the database object is valid
    Call CheckDB

    ' Retrieve the next constraint identifier using the
    ' sequence class
    Set mConstraintSeq = New cSequence
    Set mConstraintSeq.IdDatabase = mdbsConstraintDB
    mConstraintSeq.IdentifierColumn = "constraint_id"
    lngNextId = mConstraintSeq.Identifier
    Set mConstraintSeq = Nothing

    NextIdentifier = lngNextId
    Exit Property

NextIdentifierErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "NextIdentifier"
    On Error GoTo 0
```

```
    Err.Raise vbObjectError + errStepIdGetFailed, _
        mstrSource, LoadResString(errStepIdGetFailed)

End Property

Private Sub CheckDB()
    ' Check if the database object has been initialized

    If mdbsConstraintDB Is Nothing Then
        ShowError errInvalidDB
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidDB, _
            mstrModuleName, LoadResString(errInvalidDB)
    End If

End Sub

Public Sub Delete()
    ' Deletes the step constraint record from the database

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr
    mstrSource = mstrModuleName & "Delete"

    ' There can be multiple constraints for a step,
    ' meaning that there can be multiple constraint records
    ' with the same constraint_id. Only a combination
    ' of the step_id, version and constraint_id will be
    ' unique
    strDelete = "delete from step_constraints " & _
            " where constraint_id = [cons_id]" & _
            " and step_id = [s_id] " & _
            " and version_no = [ver_no] "
    Set qy = mdbsConstraintDB.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

'    strDelete = "Delete from step_constraints " & _
'            " where constraint_id = " & Str(mlngConstraintId) & _
'            " and step_id = " & Str(mlngStepId) & _
'            " and version_no = " & mstrSQ & mstrVersionNo & mstrSQ
'
'    'BugMessage strDelete
'    mdbsConstraintDB.Execute strDelete, dbFailOnError

    Exit Sub

DeleteErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteConstraintFailed, _
            mstrSource, _
            LoadResString(errDeleteConstraintFailed)
End Sub
Public Sub Modify()
    ' Updates the sequence no of the step constraint record
    ' in the database

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo Modify

    ' First check if the database object is valid
    Call CheckDB

    ' Any record validations
    Call Validate

    ' There can be multiple constraints for a step,
    ' meaning that there can be multiple constraint records
    ' with the same constraint_id. Only a combination
    ' of the step_id, version and constraint_id will be
    ' unique
    ' Create a temporary querydef object with the modify string
    strUpdate = "Update step_constraints " & _
            " set sequence_no = [seq_no] " & _
            " where constraint_id = [cons_id] " & _
            " and step_id = [s_id] " & _
            " and version_no = [ver_no] "
    Set qy = mdbsConstraintDB.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the parameter values to the
    ' querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

'    strUpdate = "Update step_constraints " & _
'            " set sequence_no = " & Str(mintSequenceNo) & _
'            " where constraint_id = " & Str(mlngConstraintId) & _
'            " and step_id = " & Str(mlngStepId) & _
'            " and version_no = " & mstrSQ & mstrVersionNo & mstrSQ
'
'    'BugMessage strUpdate
'    mdbsConstraintDB.Execute strUpdate, dbFailOnError
    Exit Sub

Modify:
    LogErrors Errors
    mstrSource = mstrModuleName & "Modify"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateConstraintFailed, _
            mstrSource, _
            LoadResString(errUpdateConstraintFailed)
```

```
End Sub
Public Property Get Position() As Long

    Position = mlngPosition

End Property
Public Property Let Position(ByVal RHS As Long)

    mlngPosition = RHS

End Property

Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    ' No validations are necessary for the constraint object

End Sub

Public Property Set NodeDB(vdata As Database)

    Set mdbsConstraintDB = vdata

End Property

Public Property Get NodeDB() As Database

    Set NodeDB = mdbsConstraintDB

End Property

Public Property Get GlobalVersionNo() As String

    GlobalVersionNo = mstrGlobalVersionNo

End Property

Public Property Let GlobalVersionNo(ByVal vdata As String)

    mstrGlobalVersionNo = vdata

End Property

Public Property Get GlobalStepId() As Long

    GlobalStepId = mlngGlobalStepId

End Property

Public Property Get ConstraintId() As Long

    ConstraintId = mlngConstraintId

End Property

Public Property Get VersionNo() As String

    VersionNo = mstrVersionNo

End Property

Public Property Get StepId() As Long

    StepId = mlngStepId

End Property

Public Property Let VersionNo(ByVal vdata As String)

    mstrVersionNo = vdata

End Property

Public Property Let StepId(ByVal vdata As Long)

    mlngStepId = vdata

End Property

Public Property Let ConstraintId(ByVal vdata As Long)
    On Error GoTo ConstraintIdErr
    mstrSource = mstrModuleName & "ConstraintId"

    If (vdata > 0) Then
        mlngConstraintId = vdata
    Else
        ' Propogate this error back to the caller
        On Error GoTo 0
        Err.Raise vbObjectError + errConstraintIdInvalid, _
            mstrSource, LoadResString(errConstraintIdInvalid)
    End If

    Exit Property

ConstraintIdErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "ConstraintId"
    On Error GoTo 0
    Err.Raise vbObjectError + errConstraintIdSetFailed, _
        mstrSource, LoadResString(errConstraintIdSetFailed)

End Property

Public Property Let GlobalStepId(ByVal vdata As Long)

    On Error GoTo GlobalStepIdErr
    mstrSource = mstrModuleName & "GlobalStepId"
```

```
    If (vdata > 0) Then
        mlngGlobalStepId = vdata
    Else
        ' Propogate this error back to the caller
        On Error GoTo 0
        Err.Raise vbObjectError + errGlobalStepIdInvalid, _
            mstrSource, LoadResString(errGlobalStepIdInvalid)
    End If

    Exit Property

GlobalStepIdErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "GlobalStepId"
    On Error GoTo 0
    Err.Raise vbObjectError + errGlobalStepIdSetFailed, _
        mstrSource, LoadResString(errGlobalStepIdSetFailed)

End Property

Public Property Let ConstraintType(ByVal vdata As ConstraintType)

    On Error GoTo ConstraintTypeErr

    ' A global step can be either a pre- or a post-execution step.
    ' These constants have been defined in the enumeration,
    ' ConstraintType, which is exposed
    Select Case vdata
        Case gintPreStep, gintPostStep
            mintConstraintType = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errConstraintTypeInvalid, _
                mstrSource, LoadResString(errConstraintTypeInvalid)
    End Select

    Exit Property

ConstraintTypeErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "ConstraintType"
    On Error GoTo 0
    Err.Raise vbObjectError + errConstraintTypeLetFailed, _
        mstrSource, LoadResString(errConstraintTypeLetFailed)

End Property

Public Property Get ConstraintType() As ConstraintType

    ConstraintType = mintConstraintType

End Property

Private Sub Class_Initialize()

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

End Sub
```

cFailedStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cFailedStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cFailedStep.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Properties of a step execution failure.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public InstanceId As Long
Public StepId As Long
Public ParentStepId As Long
Public ContCriteria As ContinuationCriteria
Public EndTime As Currency
Public AskResponse As Long
```

cFailedSteps.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cFailedSteps"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cFailedSteps.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module encapsulates a collection of failed steps.
It
```

```
'               also determines whether sub-steps of a passed in step
need
'               to be skipped due to a failure.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcFailedSteps As cVector
Public Function ExecuteSubStep(lParentStepId As Long) As Boolean
    ' Returns False if there is any condition that prevents sub-steps of
the passed
    ' in instance from being executed
    Dim lIndex As Long

    ExecuteSubStep = True

    For lIndex = 0 To Count() - 1
        If mcFailedSteps(lIndex).ContCriteria =
gintOnFailureCompleteSiblings And _
                lParentStepId <> mcFailedSteps(lIndex).ParentStepId Then
            ExecuteSubStep = False
            Exit For
        End If

        If mcFailedSteps(lIndex).ContCriteria =
gintOnFailureAbortSiblings And _
                lParentStepId = mcFailedSteps(lIndex).ParentStepId Then
            ExecuteSubStep = False
            Exit For
        End If

        If mcFailedSteps(lIndex).ContCriteria =
gintOnFailureSkipSiblings And _
                lParentStepId = mcFailedSteps(lIndex).ParentStepId Then
            ExecuteSubStep = False
            Exit For
        End If

        If mcFailedSteps(lIndex).ContCriteria = gintOnFailureAbort Then
            ExecuteSubStep = False
            Exit For
        End If

    Next lIndex

End Function
Public Sub Add(ByVal objItem As cFailedStep)

    mcFailedSteps.Add objItem

End Sub
Public Function Delete(ByVal lPosition As Long) As cFailedStep

    Set Delete = mcFailedSteps.Delete(lPosition)

End Function
Public Sub Clear()

    mcFailedSteps.Clear

End Sub
Public Function Count() As Long

    Count = mcFailedSteps.Count

End Function
Public Property Get Item(ByVal Position As Long) As cFailedStep
Attribute Item.VB_UserMemId = 0

    Set Item = mcFailedSteps.Item(Position)

End Property

Public Function StepFailed(lStepId As Long) As Boolean

    ' Returns True if a failure record already exists for the passed in
step
    Dim lIndex As Long

    StepFailed = False

    For lIndex = 0 To Count() - 1
        If mcFailedSteps(lIndex).StepId = lStepId Then
            StepFailed = True
            Exit For
        End If
    Next lIndex

End Function

Private Sub Class_Initialize()

    Set mcFailedSteps = New cVector

End Sub

Private Sub Class_Terminate()

    Set mcFailedSteps = Nothing

End Sub
```

cFileInfo.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cFileInfo"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
```

```
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cFileInfo.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     File Properties viz. name, handle, etc.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mstrFileName As String
Private mintFileHandle As Integer
Private mdbsNodeDb As Database ' Since it is used to form a
cNodeCollection
Private mlngPosition As Long ' Since it is used to form a
cNodeCollection
Public Property Get FileName() As String

    FileName = mstrFileName

End Property
Public Property Let FileName(ByVal vdata As String)

    mstrFileName = vdata

End Property
Public Property Let FileHandle(ByVal vdata As Integer)

    mintFileHandle = vdata

End Property
Public Property Set NodeDB(vdata As Database)

    Set mdbsNodeDb = vdata

End Property

Public Property Get NodeDB() As Database

    Set NodeDB = mdbsNodeDb

End Property
Public Property Get Position() As Long

    Position = mlngPosition

End Property
Public Property Let Position(ByVal vdata As Long)

    mlngPosition = vdata

End Property
Public Property Get FileHandle() As Integer

    FileHandle = mintFileHandle

End Property
```

cFileSM.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cFileSM"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Attribute VB_Ext_KEY = "SavedWithClassBuilder" ,"Yes"
Attribute VB_Ext_KEY = "Top_Level" ,"Yes"
'  FILE:        cFileSM.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates functions to open a file and write to it.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cFileSM."
Private mstrSource As String

Private mstrFileName As String
Private mintHFile As Integer
Private mstrFileHeader As String
Private mstrProjectName As String

Public Sub CloseFile()

    ' Close the file
    If mintHFile > 0 Then
        Call CloseFileSM(mstrFileName)
        mintHFile = 0
    End If

End Sub

Public Property Let ProjectName(ByVal vdata As String)
    ' An optional field - will be appended to the file
    ' header string if specified

    Const strProjectHdr As String = "Project Name:"
```

```
    mstrProjectName = vdata
    mstrFileHeader = mstrFileHeader & _
            Space$(1) & strProjectHdr & Space$(1) & _
            gstrSQ & vdata & gstrSQ

End Property
Public Property Get ProjectName() As String

    ProjectName = mstrProjectName

End Property
Public Property Get FileName() As String

    FileName = mstrFileName

End Property
Public Property Let FileName(ByVal vdata As String)

    mstrFileName = vdata

End Property
Public Sub WriteLine(strMsg As String)

    ' Writes the passed in string to the file
    Call WriteToFile(strMsg, False)

End Sub

Public Sub WriteField(strMsg As String)

    ' Writes the passed in string to the file
    Call WriteToFile(strMsg, True)

End Sub

Private Sub WriteToFile(strMsg As String, _
        blnContinue As Boolean)
    ' Writes the passed in string to the file - the
    ' Continue flag indicates whether the next line will
    ' be continued on the same line or printed on a new one

    On Error GoTo WriteToFileErr

    ' Open the file if it hasn't been already
    If mintHFile = 0 Then

        ' If the filename has not been initialized, do not
        ' attempt to open it
        If mstrFileName <> gstrEmptyString Then

            mintHFile = OpenFileSM(mstrFileName)

            If mintHFile = 0 Then
                ' The Open File command failed for some reason
                ' No point in trying to write the file header
            Else
                ' Print a file header, if a header string has been
                ' initialized
                If mstrFileHeader <> gstrEmptyString Then
                    Print #mintHFile,
                    Print #mintHFile, mstrFileHeader
                    Print #mintHFile,
                End If
            End If
        End If
    End If

    If mintHFile <> 0 Then
        If strMsg = gstrEmptyString Then
            Print #mintHFile,
        Else
            If blnContinue Then
                ' Write the message to the file - continue
                ' all subsequent characters on the same line
                Print #mintHFile, strMsg;
            Else
                ' Write the message to the file
                Print #mintHFile, strMsg
            End If
        End If
    Else
        ' Display the string to the user instead of
        ' trying to write it to the file
        ' This could be the project error log that we were
        ' trying to open! Play it safe and display errors - do
        ' not try to log them.
        MsgBox strMsg, vbOKOnly
    End If

    Exit Sub

WriteToFileErr:
    ' Log the error code raised by Visual Basic
    Call DisplayErrors(Errors)

    ' Display the string to the user instead of
    ' trying to write it to the file
    MsgBox strMsg, vbOKOnly

End Sub
Public Property Let FileHeader(ByVal vdata As String)

    mstrFileHeader = vdata

End Property
Public Property Get FileHeader() As String

    FileHeader = mstrFileHeader

End Property

Private Sub Class_Terminate()
```

```
    ' Close the file opened by this instance
    Call CloseFile

End Sub
```

cGlobalStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cGlobalStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cGlobalStep.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a global step.
'               Implements the cStep class - carries out initializations
'               and validations that are specific to global steps.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Implements cStep

' Object variable to keep the reference in
Private mcStep As cStep

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cGlobalStep."

Private Sub cStep_AddAllIterators()

    Call mcStep.AddAllIterators

End Sub

Private Sub cStep_AddIterator(cItRecord As cIterator)

    Call mcStep.AddIterator(cItRecord)

End Sub

Private Property Let cStep_ArchivedFlag(ByVal RHS As Boolean)

    mcStep.ArchivedFlag = RHS

End Property

Private Property Get cStep_ArchivedFlag() As Boolean

    cStep_ArchivedFlag = mcStep.ArchivedFlag

End Property

Private Sub Class_Initialize()

    ' Create the object
    Set mcStep = New cStep

    ' Initialize the object with valid values for a global step
    ' The global flag should be the first field to be initialized
    ' since subsequent validations might try to check if the
    ' step being created is global
    mcStep.GlobalFlag = True
    mcStep.StepType = gintGlobalStep

    ' A global step cannot have any sub-steps associated with it
    ' Hence, it will always be at Step Level 0
    mcStep.ParentStepId = 0
    mcStep.ParentVersionNo = gstrMinVersion
    mcStep.StepLevel = 0

    ' The enabled flag must be False for all global steps
    ' Global steps can be of two types
    ' a. Those that are run globally within a workspace either
    '    before every step, after every step or during the entire
    '    run, depending on the global run method
    ' b. Those that are not run globally, but qualify to be either
    '    pre or post-execution steps for other steps in the workspace.
    '    Whether or not such a step will be executed depends on
    '    whether the step for which it is defined as a pre/post
    '    step will be executed
    mcStep.EnabledFlag = False

    mcStep.ContinuationCriteria = gintNoOption
    mcStep.DegreeParallelism = gstrGlobalParallelism

End Sub
Private Sub Class_Terminate()

    ' Remove the step object
    Set mcStep = Nothing

End Sub

Private Sub cStep_Add()

    ' Call a private procedure to see if the step text has been
    ' entered - since a global step actually executes a step, entry
    ' of the text is mandatory
    Call StepTextOrFileEntered
```

```
    ' Call the Add method of the step class to carry out the insert
    mcStep.Add

End Sub

Private Function cStep_Clone(Optional cCloneStep As cStep) As cStep

    Dim cNewGlobal As cGlobalStep

    Set cNewGlobal = New cGlobalStep
    Set cStep_Clone = mcStep.Clone(cNewGlobal)

End Function


Private Property Get cStep_ContinuationCriteria() As
ContinuationCriteria

    cStep_ContinuationCriteria = mcStep.ContinuationCriteria

End Property

Private Property Let cStep_ContinuationCriteria(ByVal RHS As
ContinuationCriteria)

    ' The continuation criteria field will always be empty for a
    ' global step
    mcStep.ContinuationCriteria = 0

End Property

Private Property Let cStep_DegreeParallelism(ByVal RHS As String)

    ' Will always be zero for a global step
    mcStep.DegreeParallelism = gstrGlobalParallelism

End Property

Private Property Get cStep_DegreeParallelism() As String

    cStep_DegreeParallelism = mcStep.DegreeParallelism

End Property

Private Sub cStep_DeleteIterator(cItRecord As cIterator)

    Call mcStep.DeleteIterator(cItRecord)

End Sub

Private Sub cStep_Delete()

    mcStep.Delete

End Sub

Private Property Get cStep_EnabledFlag() As Boolean

    cStep_EnabledFlag = mcStep.EnabledFlag

End Property

Private Property Let cStep_EnabledFlag(ByVal RHS As Boolean)

    ' The enabled flag must be False for all global steps
    ' Global steps can be of two types
    ' a. Those that are run globally within a workspace either
    '    before every step, after every step or during the entire
    '    run, depending on the global run method
    ' b. Those that are not run globally, but qualify to be either
    '    pre or post-execution steps for other steps in the workspace.
    '    Whether or not such a step will be executed depends on
    '    whether the step for which it is defined as a pre/post
    '    step will be executed
    mcStep.EnabledFlag = False

End Property

Private Property Let cStep_ErrorFile(ByVal RHS As String)

    mcStep.ErrorFile = RHS

End Property

Private Property Get cStep_ErrorFile() As String

    cStep_ErrorFile = mcStep.ErrorFile

End Property

Private Property Let cStep_ExecutionMechanism(ByVal RHS As
ExecutionMethod)

    ' Whether or not the Execution Mechanism is valid will be
    ' checked by the Step class
    mcStep.ExecutionMechanism = RHS

End Property

Private Property Get cStep_ExecutionMechanism() As ExecutionMethod

    cStep_ExecutionMechanism = mcStep.ExecutionMechanism

End Property

Private Property Let cStep_FailureDetails(ByVal RHS As String)

    ' Whether or not the Failure Details are valid for the
    ' selected failure criteria will be checked by the Step class
    mcStep.FailureDetails = RHS
```

```
End Property

Private Property Get cStep_FailureDetails() As String

    cStep_FailureDetails = mcStep.FailureDetails

End Property

Private Property Get cStep_GlobalFlag() As Boolean

    cStep_GlobalFlag = mcStep.GlobalFlag

End Property

Private Property Let cStep_GlobalFlag(ByVal RHS As Boolean)

    ' Set the global flag to true
    mcStep.GlobalFlag = True

End Property

Private Function cStep_IncVersionX() As String

    cStep_IncVersionX = mcStep.IncVersionX

End Function

Private Function cStep_IncVersionY() As String

    cStep_IncVersionY = mcStep.IncVersionY

End Function


'Private Property Let cStep_GlobalRunMethod(ByVal RHS As Integer)
'
'    ' Whether or not the Global Run Method is valid for the step
'    ' will be checked by the Step class
'    mcStep.GlobalRunMethod = RHS
'
'End Property
'
'Private Property Get cStep_GlobalRunMethod() As Integer
'
'    cStep_GlobalRunMethod = mcStep.GlobalRunMethod
'
'End Property
'
Private Property Get cStep_IndOperation() As Operation

    cStep_IndOperation = mcStep.IndOperation

End Property

Private Property Let cStep_IndOperation(ByVal RHS As Operation)

    mcStep.IndOperation = RHS

End Property

Private Sub cStep_InsertIterator(cItRecord As cIterator)

    Call mcStep.InsertIterator(cItRecord)

End Sub

Private Function cStep_IsNewVersion() As Boolean
    cStep_IsNewVersion = mcStep.IsNewVersion
End Function

Private Function cStep_IteratorCount() As Long

    cStep_IteratorCount = mcStep.IteratorCount

End Function

Private Property Let cStep_IteratorName(ByVal RHS As String)

    mcStep.IteratorName = RHS

End Property

Private Property Get cStep_IteratorName() As String

    cStep_IteratorName = mcStep.IteratorName

End Property

Private Function cStep_Iterators() As Variant

    cStep_Iterators = mcStep.Iterators

End Function

Private Sub cStep_LoadIterator(cItRecord As cIterator)

    Call mcStep.LoadIterator(cItRecord)

End Sub

'Private Property Let cStep_LogFile(ByVal RHS As String)
'
'    mcStep.LogFile = RHS
'
'End Property
'
'Private Property Get cStep_LogFile() As String
'
'    cStep_LogFile = mcStep.LogFile
'
'End Property
```

```
Private Sub cStep_ModifyIterator(cItRecord As cIterator)

    Call mcStep.ModifyIterator(cItRecord)

End Sub

Private Sub cStep_Modify()

    ' Call a private procedure to see if the step text has been
    ' entered - since a global step actually executes a step,
    ' entry of the text is mandatory
    Call StepTextOrFileEntered

    ' Call the Modify method of the step class to carry out the update
    mcStep.Modify

End Sub

Private Property Get cStep_NextStepId() As Long

    cStep_NextStepId = mcStep.NextStepId

End Property

Private Property Set cStep_NodeDB(RHS As DAO.Database)

    Set mcStep.NodeDB = RHS

End Property

Private Property Get cStep_NodeDB() As DAO.Database

    Set cStep_NodeDB = mcStep.NodeDB

End Property

Private Function cStep_OldVersionNo() As String
    cStep_OldVersionNo = mcStep.OldVersionNo
End Function

Private Property Let cStep_OutputFile(ByVal RHS As String)

    mcStep.OutputFile = RHS

End Property

Private Property Get cStep_OutputFile() As String

    cStep_OutputFile = mcStep.OutputFile

End Property

Private Property Let cStep_ParentStepId(ByVal RHS As Long)

    ' A global step cannot have any sub-steps associated with it
    ' Hence, the parent step id and parent version number will be zero
    mcStep.ParentStepId = 0

End Property

Private Property Get cStep_ParentStepId() As Long

    cStep_ParentStepId = mcStep.ParentStepId

End Property

Private Property Let cStep_ParentVersionNo(ByVal RHS As String)

    ' A global step cannot have any sub-steps associated with it
    ' Hence, the parent step id and parent version number will be zero
    mcStep.ParentVersionNo = gstrMinVersion

End Property

Private Property Get cStep_ParentVersionNo() As String

    cStep_ParentVersionNo = mcStep.ParentVersionNo

End Property

Private Property Let cStep_Position(ByVal RHS As Long)

    mcStep.Position = RHS

End Property

Private Property Get cStep_Position() As Long

    cStep_Position = mcStep.Position

End Property

Private Sub cStep_RemoveIterator(cItRecord As cIterator)

    Call mcStep.RemoveIterator(cItRecord)

End Sub

Private Sub cStep_SaveIterators()

    Call mcStep.SaveIterators

End Sub

Private Property Let cStep_SequenceNo(ByVal RHS As Integer)

    mcStep.SequenceNo = RHS

End Property

Private Property Get cStep_SequenceNo() As Integer
```

```
    cStep_SequenceNo = mcStep.SequenceNo

End Property

Private Property Let cStep_StepId(ByVal RHS As Long)

    mcStep.StepId = RHS

End Property
Private Property Get cStep_StepId() As Long

    cStep_StepId = mcStep.StepId

End Property


Private Property Let cStep_StepLabel(ByVal RHS As String)

    mcStep.StepLabel = RHS

End Property

Private Property Get cStep_StepLabel() As String

    cStep_StepLabel = mcStep.StepLabel

End Property

Private Property Let cStep_StartDir(ByVal RHS As String)

    mcStep.StartDir = RHS

End Property

Private Property Get cStep_StartDir() As String

    cStep_StartDir = mcStep.StartDir

End Property

Private Property Let cStep_StepLevel(ByVal RHS As Integer)

    ' A global step cannot have any sub-steps associated with it
    ' Hence, it will always be at step level 0
    mcStep.StepLevel = 0

End Property

Private Property Get cStep_StepLevel() As Integer

    cStep_StepLevel = mcStep.StepLevel

End Property

Private Property Let cStep_StepText(ByVal RHS As String)

    mcStep.StepText = RHS

End Property

Private Property Get cStep_StepText() As String

    cStep_StepText = mcStep.StepText

End Property

Private Property Let cStep_StepTextFile(ByVal RHS As String)

    mcStep.StepTextFile = RHS

End Property

Private Property Get cStep_StepTextFile() As String

    cStep_StepTextFile = mcStep.StepTextFile

End Property

Private Property Let cStep_StepType(RHS As gintStepType)

    mcStep.StepType = gintGlobalStep

End Property

Private Property Get cStep_StepType() As gintStepType

    cStep_StepType = mcStep.StepType

End Property

Private Sub cStep_UnloadIterators()

    Call mcStep.UnloadIterators

End Sub

Private Sub cStep_UpdateIterator(cItRecord As cIterator)

    Call mcStep.UpdateIterator(cItRecord)

End Sub

Private Sub cStep_UpdateIteratorVersion()

    Call mcStep.UpdateIteratorVersion

End Sub

Private Sub cStep_Validate()
    ' The validate routines for each of the steps will
    ' carry out the specific validations for the type and
    ' call the generic validation routine
```

```
    On Error GoTo cStep_ValidateErr
    mstrSource = mstrModuleName & "cStep_Validate"

    ' Validations specific to global steps

    ' Check if the step text or a file name has been
    ' specified
    Call StepTextOrFileEntered

    ' The step level must be zero for all globals
    If mcStep.StepLevel <> 0 Then
        ShowError errStepLevelZeroForGlobal
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.EnabledFlag Then
        ShowError errEnabledFlagFalseForGlobal
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.DegreeParallelism > 0 Then
        ShowError errDegParallelismNullForGlobal
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.ContinuationCriteria > 0 Then
        ShowError errContCriteriaNullForGlobal
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    mcStep.Validate

    Exit Sub

cStep_ValidateErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "cStep_Validate"
    On Error GoTo 0
    Err.Raise vbObjectError + errValidateFailed, _
            mstrSource, _
            LoadResString(errValidateFailed)
End Sub
Private Sub StepTextOrFileEntered()

    ' Checks if either the step text or the name of the file containing
    ' the text has been entered
    ' If both of them are null or both of them are not null,
    ' the global step is invalid and an error is raised

    If StringEmpty(mcStep.StepText) And StringEmpty(mcStep.StepTextFile)
Then
        ShowError errStepTextAndFileNull
        On Error GoTo 0
        Err.Raise vbObjectError + errStepTextAndFileNull, _
            mstrSource, LoadResString(errStepTextAndFileNull)
    ElseIf Not StringEmpty(mcStep.StepText) And Not
StringEmpty(mcStep.StepTextFile) Then
        ShowError errStepTextOrFile
        On Error GoTo 0
        Err.Raise vbObjectError + errStepTextOrFile, _
            mstrSource, LoadResString(errStepTextOrFile)
    End If

End Sub

Private Property Let cStep_VersionNo(ByVal RHS As String)

    mcStep.VersionNo = RHS

End Property

Private Property Get cStep_VersionNo() As String

    cStep_VersionNo = mcStep.VersionNo

End Property

Private Property Let cStep_WorkspaceId(ByVal RHS As Long)

    mcStep.WorkspaceId = RHS

End Property

Private Property Get cStep_WorkspaceId() As Long

    cStep_WorkspaceId = mcStep.WorkspaceId

End Property
```

cInstance.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cInstance"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
```

```
'  FILE:        cInstance.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of an instance.
'               An instance is created when a step is executed for a
'               particular iterator value (if applicable) at 'run' time.
'               Contains functions to determine if an instance is
running,
'               complete, and so on.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cInstance."
Private mstrSource As String

Private mcStep As cStep
Public Key As String ' Node key for the step being executed
Public InstanceId As Long
Public ParentInstanceId As Long ' The parent instance
Private mblnNoMoreToStart As Boolean
Private mblnComplete As Boolean
Public StartTime As Currency
Public EndTime As Currency
Public ElapsedTime As Currency
Private mintStatus As InstanceStatus
Public DegreeParallelism As Integer
Private mcIterators As cRunColIt

' A collection of all the sub-steps for this step
Private mcSubSteps As cSubSteps
Public Sub UpdateStartTime(lStepId As Long, Optional ByVal StartTm As
Currency = gdtmEmpty, _
        Optional ByVal EndTm As Currency = gdtmEmpty, _
        Optional ByVal Elapsed As Currency = 0)
    ' We do not maintain start and end timestamps for the constraint
    ' of a step. Hence we check if the process that just started/
    ' terminated is the worker step that is being executed. If so,
    ' we update the start/end time and status on the instance record.

    BugAssert (StartTm <> gdtmEmpty) Or (EndTm <> gdtmEmpty), "Mandatory
parameter missing."

    ' Make sure that we are executing the actual step and not
    ' a pre or post-execution constraint
    If mcStep.StepId = lStepId Then
        If StartTm <> 0 Then
            StartTime = StartTm
            mintStatus = gintRunning
        Else
            EndTime = EndTm
            ElapsedTime = Elapsed
            mintStatus = gintComplete
        End If
    End If

End Sub
Public Function ValidForIteration(cParentInstance As cInstance, _
        ByVal intConsType As ConstraintType) As Boolean
    ' Returns true if the instance passed in is the first or
    ' last iteration for the step, depending on the constraint type

    Dim cSubStepRec As cSubStep
    Dim vntIterators As Variant

    On Error GoTo ValidForIterationErr

    If cParentInstance Is Nothing Then
        ' This will only be true for the dummy instance, which
        ' cannot have any iterators defined for it
        ValidForIteration = True
        Exit Function
    End If

    vntIterators = mcStep.Iterators

    If Not StringEmpty(mcStep.IteratorName) And Not
IsEmpty(vntIterators) Then

        Set cSubStepRec = cParentInstance.QuerySubStep(mcStep.StepId)

        If intConsType = gintPreStep Then
            ' Pre-execution constraints will only be executed
            ' before the first iteration
            If cSubStepRec.LastIterator.IteratorType = gintValue Then
                ValidForIteration = (cSubStepRec.LastIterator.Sequence =
_
                        gintMinIteratorSequence)
            Else
                ValidForIteration = (cSubStepRec.LastIterator.Value = _
                        cSubStepRec.LastIterator.RangeFrom)
            End If
        Else
            ' Post-execution constraints will only be executed
            ' after the last iteration - check if there are any
            ' pending iterations
            ValidForIteration = cSubStepRec.NextIteration(mcStep) Is
Nothing
        End If
    Else
        ValidForIteration = True
    End If

    Exit Function

ValidForIterationErr:
    ' Log the error code raised by Visual Basic
```

```
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "ValidForIteration"
    Err.Raise vbObjectError + errExecInstanceFailed, _
            mstrSource, LoadResString(errExecInstanceFailed)

End Function

Public Sub CreateSubStep(cSubStepDtls As cStep, RunParams As
cArrParameters)

    Dim cNewSubStep As cSubStep

    On Error GoTo CreateSubStepErr

    Set cNewSubStep = New cSubStep

    cNewSubStep.StepId = cSubStepDtls.StepId
    cNewSubStep.TasksComplete = 0
    cNewSubStep.TasksRunning = 0

    ' Initialize the iterator for the instance
    Set cNewSubStep.LastIterator = New cRunItDetails
    Call cNewSubStep.InitializeIt(cSubStepDtls, RunParams)

    ' Add add the substep to the collection
    mcSubSteps.Add cNewSubStep

    Set cNewSubStep = Nothing

    Exit Sub

CreateSubStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "CreateSubStep"
    Err.Raise vbObjectError + errProgramError, mstrSource, _
        LoadResString(errProgramError)

End Sub

Public Function QuerySubStep(ByVal SubStepId As Long) As cSubStep
    ' Retrieves the sub-step record for the passed in sub-step id

    Dim lngIndex As Long

    On Error GoTo QuerySubStepErr

    ' Find the sub-step node with the matching step id
    For lngIndex = 0 To mcSubSteps.Count - 1
        If mcSubSteps(lngIndex).StepId = SubStepId Then
            Set QuerySubStep = mcSubSteps(lngIndex)
            Exit For
        End If
    Next lngIndex

    Exit Function

QuerySubStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "QuerySubStep"
    Err.Raise vbObjectError + errNavInstancesFailed, _
            mstrSource, LoadResString(errNavInstancesFailed)

End Function
Public Property Let AllStarted(ByVal vdata As Boolean)

    'bugmessage "Set All Started to " & vData & " for : " & _
        mstrKey

    mblnNoMoreToStart = vdata

End Property
Public Property Get AllStarted() As Boolean

    AllStarted = mblnNoMoreToStart

End Property
Public Property Let AllComplete(ByVal vdata As Boolean)

    'bugmessage "Set All Complete to " & vData & " for : " & _
        mstrKey

    mblnComplete = vdata

End Property

Public Property Get AllComplete() As Boolean

    AllComplete = mblnComplete

End Property

Public Sub ChildExecuted(mlngStepId As Long)
    ' This procedure is called when a sub-step executes.

    Dim lngIndex As Long

    On Error GoTo ChildExecutedErr

    BugAssert mcStep.StepType = gintManagerStep

    For lngIndex = 0 To mcSubSteps.Count - 1
        If mcSubSteps(lngIndex).StepId = mlngStepId Then
            mcSubSteps(lngIndex).TasksRunning = _
                    mcSubSteps(lngIndex).TasksRunning + 1
'           BugMessage "Tasks Running for Step Id : " & _
'                   CStr(mcSubSteps(lngIndex).StepId) & _
'                   " Instance Id: " & InstanceId & _
```

```
'                   " = " & mcSubSteps(lngIndex).TasksRunning
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcSubSteps.Count - 1 Then
        ' The child step wasn't found - raise an error
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidChild, mstrModuleName, _
                LoadResString(errInvalidChild)
    End If

    Exit Sub

ChildExecutedErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInstanceOpFailed, mstrModuleName &
"ChildExecuted", _
            LoadResString(errInstanceOpFailed)

End Sub
Public Sub ChildTerminated(mlngStepId As Long)
    ' This procedure is called when any sub-step process
    ' terminates. Note: The TasksComplete field will be
    ' updated only when all the instances for a sub-step
    ' complete execution.
    Dim lngIndex As Long

    On Error GoTo ChildTerminatedErr

    BugAssert mcStep.StepType = gintManagerStep

    For lngIndex = 0 To mcSubSteps.Count - 1

        If mcSubSteps(lngIndex).StepId = mlngStepId Then
            mcSubSteps(lngIndex).TasksRunning = _
                    mcSubSteps(lngIndex).TasksRunning - 1
'            BugMessage "Tasks Running for Step Id : " & _
'                    CStr(mcSubSteps(lngIndex).StepId) & _
'                    " Instance Id: " & InstanceId & _
'                    " = " & mcSubSteps(lngIndex).TasksRunning

            BugAssert mcSubSteps(lngIndex).TasksRunning >= 0, _
                    "Tasks running for " & CStr(mlngStepId) & _
                    " Instance Id " & InstanceId & " is less than 0."
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcSubSteps.Count - 1 Then
        ' The child step wasn't found - raise an error
        On Error GoTo 0
        Err.Raise errInvalidChild, mstrModuleName & "ChildTerminated", _
                LoadResString(errInvalidChild)
    End If

    Exit Sub

ChildTerminatedErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "ChildTerminated"
    Err.Raise vbObjectError + errInstanceOpFailed, mstrSource, _
            LoadResString(errInstanceOpFailed)

End Sub
Public Sub ChildCompleted(mlngStepId As Long)
    ' This procedure is called when any a sub-step completes
    ' execution. Note: The TasksComplete field will be
    ' incremented.
    Dim lngIndex As Long

    On Error GoTo ChildCompletedErr

    BugAssert mcStep.StepType = gintManagerStep

    For lngIndex = 0 To mcSubSteps.Count - 1
        BugAssert mcSubSteps(lngIndex).TasksComplete >= 0, _
                "Tasks complete for " &
CStr(mcSubSteps(lngIndex).StepId) & _
                " Instance Id " & InstanceId & " is less than 0."

        If mcSubSteps(lngIndex).StepId = mlngStepId Then
            mcSubSteps(lngIndex).TasksComplete = _
                    mcSubSteps(lngIndex).TasksComplete + 1
'            BugMessage "Tasks Complete for Step Id : " & _
'                    CStr(mcSubSteps(lngIndex).StepId) & _
'                    " Instance Id: " & InstanceId & _
'                    " = " & mcSubSteps(lngIndex).TasksComplete
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcSubSteps.Count - 1 Then
        ' The child step wasn't found - raise an error
        On Error GoTo 0
        Err.Raise errInvalidChild, mstrModuleName, _
                LoadResString(errInvalidChild)
    End If

    Exit Sub

ChildCompletedErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInstanceOpFailed, mstrModuleName &
"ChildCompleted", _
            LoadResString(errInstanceOpFailed)
```

```
End Sub
Public Sub ChildDeleted(mlngStepId As Long)
    ' This procedure is called when a sub-step needs to be re-executed
    ' Note: The TasksComplete field is decremented. We needn't worry
about
    ' the TasksRunning field since no steps are currently running.
    Dim lngIndex As Long

    On Error GoTo ChildDeletedErr

    BugAssert mcStep.StepType = gintManagerStep

    For lngIndex = 0 To mcSubSteps.Count - 1

        If mcSubSteps(lngIndex).StepId = mlngStepId Then
            mcSubSteps(lngIndex).TasksRunning = _
                    mcSubSteps(lngIndex).TasksRunning - 1

            BugAssert mcSubSteps(lngIndex).TasksRunning >= 0, _
                    "Tasks running for " &
CStr(mcSubSteps(lngIndex).StepId) & _
                    " Instance Id " & InstanceId & " is less than 0."
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcSubSteps.Count - 1 Then
        ' The child step wasn't found - raise an error
        On Error GoTo 0
        Err.Raise errInvalidChild, mstrModuleName, _
                LoadResString(errInvalidChild)
    End If

    Exit Sub

ChildDeletedErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInstanceOpFailed, mstrModuleName &
"ChildDeleted", _
            LoadResString(errInstanceOpFailed)

End Sub
Private Sub RaiseErrForWorker()

    If mcStep.StepType <> gintManagerStep Then
        On Error GoTo 0
        mstrSource = mstrModuleName & "RaiseErrForWorker"
        Err.Raise vbObjectError + errInvalidForWorker, _
                mstrSource, _
                LoadResString(errInvalidForWorker)
    End If

End Sub
Public Property Get Step() As cStep

    Set Step = mcStep

End Property
Public Property Get Iterators() As cRunColIt

    Set Iterators = mcIterators

End Property
Public Property Get SubSteps() As cSubSteps

    Call RaiseErrForWorker

    Set SubSteps = mcSubSteps

End Property
Public Property Set Step(cRunStep As cStep)

    Set mcStep = cRunStep

End Property

Public Property Set Iterators(cIts As cRunColIt)

    Set mcIterators = cIts

End Property
Public Property Get IsPending() As Boolean
    ' Returns true if the step has any substeps that need
    ' execution
    Dim lngIndex As Long
    Dim lngRunning As Long

    Call RaiseErrForWorker

    If Not mblnComplete And Not mblnNoMoreToStart Then
        ' Get a count of all the substeps that are already being
        ' executed
        lngRunning = 0
        For lngIndex = 0 To mcSubSteps.Count - 1
            lngRunning = lngRunning + mcSubSteps(lngIndex).TasksRunning
        Next lngIndex

        IsPending = (lngRunning < DegreeParallelism)
    Else
        ' This should be sufficient to prove that there r no
        ' more sub-steps to be executed.
        ' mblnComplete: Handles the case where all steps have
        ' been executed
        ' mblnNoMoreToStart: Handles the case where the step
        ' has a degree of parallelism greater than the total
        ' number of sub-steps available to execute
        IsPending = False
    End If
```

```
End Property
Public Property Get IsRunning() As Boolean
    ' Returns true if the any one of the substeps is still
    ' executing
    Dim lngIndex As Long

    Call RaiseErrForWorker

    IsRunning = False

    ' If a substep has no currently executing tasks and
    ' the tasks completed is greater than zero, then we can
    ' assume that it has completed execution (otherwise we
    ' would've run a new task the moment one completed!)
    For lngIndex = 0 To mcSubSteps.Count - 1
        If mcSubSteps(lngIndex).TasksRunning > 0 Then
            IsRunning = True
            Exit For
        End If
    Next lngIndex

End Property
Public Property Get TotalRunning() As Long
    ' Returns the total number of substeps that are executing
    Dim lngTotalProcesses As Long
    Dim lngIndex As Long

    Call RaiseErrForWorker

    lngTotalProcesses = 0
    For lngIndex = 0 To mcSubSteps.Count - 1
        BugAssert mcSubSteps(lngIndex).TasksRunning >= 0, _
                "Tasks running for " & CStr(mcSubSteps(lngIndex).StepId)
& _
                " is less than 0."
        lngTotalProcesses = lngTotalProcesses +
mcSubSteps(lngIndex).TasksRunning
    Next lngIndex

    TotalRunning = lngTotalProcesses
End Property
Public Property Get RunningForStep(lngSubStepId As Long) As Long
    ' Returns the total number of instances of the substep
    ' that are executing
    Dim lngIndex As Long

    Call RaiseErrForWorker

    For lngIndex = 0 To mcSubSteps.Count - 1
        BugAssert mcSubSteps(lngIndex).TasksRunning >= 0, _
                "Tasks running for " & CStr(mcSubSteps(lngIndex).StepId)
& _
                " is less than 0."

        If mcSubSteps(lngIndex).StepId = lngSubStepId Then
            RunningForStep = mcSubSteps(lngIndex).TasksRunning
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcSubSteps.Count - 1 Then
        ' The child step wasn't found - raise an error
        On Error GoTo 0
        Err.Raise errInvalidChild, mstrSource, _
                LoadResString(errInvalidChild)
    End If

End Property

Public Property Let Status(ByVal vdata As InstanceStatus)

    mintStatus = vdata

End Property

Public Property Get Status() As InstanceStatus

    Status = mintStatus

End Property
Private Sub Class_Initialize()

    Set mcSubSteps = New cSubSteps

    mblnNoMoreToStart = False
    mblnComplete = False
    StartTime = gdtmEmpty
    EndTime = gdtmEmpty

End Sub

Private Sub Class_Terminate()

    mcSubSteps.Clear
    Set mcSubSteps = Nothing

End Sub
```

clnstances.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cInstances"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cInstances.cls
'              Microsoft TPC-H Kit Ver. 1.00
```

```
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Implements a collection of cInstance objects.
'              Type-safe wrapper around cVector.
'              Also contains additional functions to query an instance,
etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cInstance."
Private mstrSource As String

Private mcInstances As cVector

Public Function QueryInstance(ByVal InstanceId As Long) As cInstance
    ' Retrieves the record for the passed in instance from
    ' the collection

    Dim lngIndex As Long

    On Error GoTo QueryInstanceErr

    ' Check for valid values of the instance id
    If InstanceId > 0 Then
        ' Find the run node with the matching step id
        For lngIndex = 0 To Count() - 1
            If mcInstances(lngIndex).InstanceId = InstanceId Then
                Set QueryInstance = mcInstances(lngIndex)
                Exit For
            End If
        Next lngIndex

        If lngIndex > mcInstances.Count - 1 Then
            On Error GoTo 0
            Err.Raise vbObjectError + errQueryFailed, mstrSource, _
                    LoadResString(errQueryFailed)
        End If
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errQueryFailed, mstrSource, _
                LoadResString(errQueryFailed)
    End If

    Exit Function

QueryInstanceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "QueryInstance"
    Err.Raise vbObjectError + errQueryFailed, _
            mstrSource, LoadResString(errQueryFailed)

End Function

Public Function QueryPendingInstance(ByVal ParentInstanceId As Long, _
        ByVal lngSubStepId As Long) As cInstance
    ' Retrieves a pending instance for the passed in substep
    ' and the given parent instance id.

    Dim lngIndex As Long

    On Error GoTo QueryPendingInstanceErr

    ' Find the run node with the matching step id
    For lngIndex = 0 To Count() - 1
        If mcInstances(lngIndex).ParentInstanceId = ParentInstanceId And
_
                mcInstances(lngIndex).Step.StepId = lngSubStepId Then
            ' Put in a separate if condition since the IsPending
            ' property is valid only for manager steps. If the
            ' calling procedure does not pass a manager step
            ' identifier, the procedure will error out.
            If mcInstances(lngIndex).IsPending Then
                Set QueryPendingInstance = mcInstances(lngIndex)
                Exit For
            End If
        End If
    Next lngIndex

    Exit Function

QueryPendingInstanceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "QueryPendingInstance"
    Err.Raise vbObjectError + errQueryFailed, _
            mstrSource, LoadResString(errQueryFailed)

End Function
Public Function InstanceAborted(cSubStepRec As cSubStep) As Boolean

    Dim lIndex As Long

    InstanceAborted = False

    For lIndex = 0 To Count() - 1
        If mcInstances(lIndex).Step.StepId = cSubStepRec.StepId And _
                mcInstances(lIndex).Status = gintAborted Then
            InstanceAborted = True
            Exit For
        End If
    Next lIndex

End Function
Public Function CompletedInstanceExists(lParentInstance As Long, _
```

```
        cSubStepDtls As cStep) As Boolean
    ' Checks if there is a completed instance of the passed in step

    Dim lngIndex As Long

    CompletedInstanceExists = False

    If cSubStepDtls.StepType = gintManagerStep Then
        ' Find the run node with the matching step id
        For lngIndex = 0 To Count() - 1
            If mcInstances(lngIndex).ParentInstanceId = lParentInstance
And _
                   mcInstances(lngIndex).Step.StepId =
cSubStepDtls.StepId Then
                ' Put in a separate if condition since the IsPending
                ' property is valid only for manager steps.
                BugAssert (Not mcInstances(lngIndex).IsPending),
"Pending instance exists!"

                CompletedInstanceExists = True
                Exit Function
            End If
        Next lngIndex
    End If

End Function
Public Sub Add(ByVal objItem As cInstance)

    mcInstances.Add objItem

End Sub


Public Sub Clear()

    mcInstances.Clear

End Sub


Public Function Count() As Long

    Count = mcInstances.Count

End Function


Public Function Delete(ByVal lngDelete As Long) As cInstance

    Set Delete = mcInstances.Delete(lngDelete)

End Function


Public Property Set Item(Optional ByVal Position As Long, _
        RHS As cInstance)
    If Position = -1 Then
        Position = 0
    End If
    Set mcInstances(Position) = RHS

End Property

Public Property Get Item(Optional ByVal Position As Long = -1) _
        As cInstance
Attribute Item.VB_UserMemId = 0

    If Position = -1 Then
        Position = 0
    End If
    Set Item = mcInstances.Item(Position)

End Property

Private Sub Class_Initialize()

    Set mcInstances = New cVector

End Sub


Private Sub Class_Terminate()

    Set mcInstances = Nothing

End Sub
```

cIterator.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cIterator"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cIterator.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of an iterator.
'               Contains functions to insert, update and delete
'               iterator_values records from the database.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
```

```
Implements cNode

' Module level variables to store the property values
Private mintType As Integer
Private mintSequenceNo As Integer
Private mstrValue As String
Private mdbsIteratorDB As Database
Private mintOperation As Integer
Private mlngPosition As Long

Private Const mstrModuleName As String = "cIterator."
Private mstrSource As String

Public Enum ValueType
    gintFrom = 1
    gintTo
    gintStep
    gintValue
End Enum
Public Property Get Value() As String

    Value = mstrValue

End Property
Public Property Let Value(ByVal vdata As String)

    mstrValue = vdata

End Property

Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property
Public Property Let IndOperation(ByVal vdata As Operation)

    On Error GoTo IndOperationErr
    mstrSource = mstrModuleName & "IndOperation"

    ' The valid operations are define in the cOperations
    ' class. Check if the operation is valid
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errInvalidOperation, _
                    mstrSource, LoadResString(errInvalidOperation)
    End Select

    Exit Property

IndOperationErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "IndOperation"
    On Error GoTo 0
    Err.Raise vbObjectError + errLetOperationFailed, _
        mstrSource, LoadResString(errLetOperationFailed)

End Property

Public Function Clone() As cIterator

    ' Creates a copy of a given Iterator

    Dim cItClone As cIterator

    On Error GoTo CloneErr

    Set cItClone = New cIterator

    ' Copy all the iterator properties to the newly
    ' created object
    cItClone.IteratorType = mintType
    cItClone.SequenceNo = mintSequenceNo
    cItClone.IndOperation = mintOperation
    cItClone.Value = mstrValue

    ' And set the return value to the newly created Iterator
    Set Clone = cItClone

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, _
            mstrSource, LoadResString(errCloneFailed)

End Function
Public Property Get SequenceNo() As Integer

    SequenceNo = mintSequenceNo

End Property

Public Property Let SequenceNo(ByVal vdata As Integer)
    mintSequenceNo = vdata
End Property

Public Sub Add(ByVal lngStepId As Long, _
        strVersion As String)
    ' Inserts a new iterator values record into the database

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddIteratorErr
```

```
    ' First check if the database object is valid
    Call CheckDB

    ' Create a temporary querydef object
    strInsert = "insert into iterator_values " & _
            "( step_id, version_no, type, " & _
            " iterator_value, sequence_no ) " & _
            " values ( [st_id], [ver_no], [it_typ], " & _
            " [it_val], [seq_no] )"
    Set qy = mdbsIteratorDB.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy, lngStepId, strVersion)

    qy.Execute dbFailOnError
    qy.Close

    Exit Sub

AddIteratorErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "AddIterator"
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertIteratorFailed, _
            mstrSource, _
            LoadResString(errInsertIteratorFailed)
End Sub

Private Sub AssignParameters(qyExec As DAO.QueryDef, _
        ByVal lngStepId As Long, _
        strVersion As String)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the field names. When the parameter names are
    ' the same as the field names, parameters in the where
    ' clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr
    mstrSource = mstrModuleName & "AssignParameters"

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[st_id]"
                prmParam.Value = lngStepId

            Case "[ver_no]"
                prmParam.Value = strVersion

            Case "[it_typ]"
                prmParam.Value = mintType

            Case "[it_val]"
                prmParam.Value = mstrValue

            Case "[seq_no]"
                prmParam.Value = mintSequenceNo

            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource, _
                        prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrSource, _
                        LoadResString(errInvalidParameter)
        End Select
    Next prmParam

    Exit Sub

AssignParametersErr:

    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrSource, LoadResString(errAssignParametersFailed)

End Sub
Private Sub CheckDB()
    ' Check if the database object has been initialized

    If mdbsIteratorDB Is Nothing Then
        ShowError errInvalidDB
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidDB, _
            mstrModuleName, LoadResString(errInvalidDB)
    End If

End Sub

Public Sub Delete(ByVal lngStepId As Long, _
        strVersion As String)
    ' Deletes the step iterator record from the database

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteIteratorErr
    mstrSource = mstrModuleName & "DeleteIterator"

    ' There can be multiple iterators for a step.
    ' However the values that an iterator for a step can
    ' assume will be unique, meaning that a combination of
    ' the iterator_id and value will be unique.
    strDelete = "delete from iterator_values " & _
            " where step_id = [st_id]" & _
            " and version_no = [ver_no]" & _
            " and iterator_value = [it_val] "
    Set qy = mdbsIteratorDB.CreateQueryDef(gstrEmptyString, strDelete)
```

```
    Call AssignParameters(qy, lngStepId, strVersion)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteIteratorErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "DeleteIterator"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteIteratorFailed, _
            mstrSource, _
            LoadResString(errDeleteIteratorFailed)
End Sub
Public Sub Update(ByVal lngStepId As Long, strVersion As String)
    ' Updates the sequence no of the step iterator record
    ' in the database

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo UpdateErr

    ' First check if the database object is valid
    Call CheckDB

    If mintType = gintValue Then
        ' If the iterator is of type value, only the sequence of the
values can get updated
        strUpdate = "Update iterator_values " & _
                " set sequence_no = [seq_no] " & _
                " where step_id = [st_id]" & _
                " and version_no = [ver_no]" & _
                " and iterator_value = [it_val] "
    Else
        ' If the iterator is of type range, only the values can get
updated
        strUpdate = "Update iterator_values " & _
                " set iterator_value = [it_val] " & _
                " where step_id = [st_id]" & _
                " and version_no = [ver_no]" & _
                " and type = [it_typ] "
    End If

    Set qy = mdbsIteratorDB.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the parameter values to the
    ' querydef object
    Call AssignParameters(qy, lngStepId, strVersion)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

UpdateErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Update"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateConstraintFailed, _
            mstrSource, _
            LoadResString(errUpdateConstraintFailed)

End Sub
Public Property Set NodeDB(vdata As Database)

    Set mdbsIteratorDB = vdata

End Property

Public Property Get NodeDB() As Database

    Set NodeDB = mdbsIteratorDB

End Property
Public Property Get Position() As Long

    Position = mlngPosition

End Property
Public Property Let Position(ByVal vdata As Long)

    mlngPosition = vdata

End Property

Public Property Let IteratorType(ByVal vdata As ValueType)

    On Error GoTo TypeErr
    mstrSource = mstrModuleName & "Type"

    ' These constants have been defined in the enumeration,
    ' Type, which is exposed
    Select Case vdata
        Case gintFrom, gintTo, gintStep, gintValue
            mintType = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errTypeInvalid, _
                mstrSource, LoadResString(errTypeInvalid)
    End Select

    Exit Property

TypeErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Type"
    On Error GoTo 0
    Err.Raise vbObjectError + errTypeInvalid, _
```

```
        mstrSource, LoadResString(errTypeInvalid)

End Property

Public Property Get IteratorType() As ValueType

    IteratorType = mintType

End Property
Public Sub Validate()

    ' No validations necessary for the iterator class

End Sub

Private Sub Class_Initialize()

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

End Sub


Private Property Let cNode_IndOperation(ByVal vdata As Operation)

    On Error GoTo IndOperationErr
    mstrSource = mstrModuleName & "IndOperation"

    ' The valid operations are define in the cOperations
    ' class. Check if the operation is valid
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errInvalidOperation, _
                    mstrSource, LoadResString(errInvalidOperation)
    End Select

    Exit Property

IndOperationErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "IndOperation"
    On Error GoTo 0
    Err.Raise vbObjectError + errLetOperationFailed, _
        mstrSource, LoadResString(errLetOperationFailed)

End Property

Private Property Get cNode_IndOperation() As Operation

    IndOperation = mintOperation

End Property

Private Property Set cNode_NodeDB(RHS As DAO.Database)

    Set mdbsIteratorDB = RHS

End Property

Private Property Get cNode_NodeDB() As DAO.Database

    Set cNode_NodeDB = mdbsIteratorDB

End Property


Private Property Let cNode_Position(ByVal vdata As Long)

    mlngPosition = vdata

End Property


Private Property Get cNode_Position() As Long

    cNode_Position = mlngPosition

End Property


Private Sub cNode_Validate()

    ' No validations necessary for the iterator class

End Sub


Private Property Let cNode_Value(ByVal vdata As String)

    mstrValue = vdata

End Property

Private Property Get cNode_Value() As String

    Value = mstrValue

End Property
```

cManager.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cManager"
```

```
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cManager.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a manager
step.
'               Implements the cStep class - carries out initializations
'               and validations that are specific to manager steps.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Implements cStep

' Object variable to keep the step reference in
Private mcStep As cStep

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cManager."
Private Sub cStep_AddAllIterators()

    Call mcStep.AddAllIterators

End Sub

Private Property Let cStep_StartDir(ByVal RHS As String)

    mcStep.StartDir = RHS

End Property

Private Property Get cStep_StartDir() As String

    cStep_StartDir = mcStep.StartDir

End Property
Private Sub cStep_Delete()

    Call mcStep.Delete

End Sub

Private Property Set cStep_NodeDB(RHS As DAO.Database)

    Set mcStep.NodeDB = RHS

End Property

Private Function cStep_IncVersionY() As String

    cStep_IncVersionY = mcStep.IncVersionY

End Function
Private Function cStep_IsNewVersion() As Boolean
    cStep_IsNewVersion = mcStep.IsNewVersion
End Function
Private Function cStep_OldVersionNo() As String
    cStep_OldVersionNo = mcStep.OldVersionNo
End Function

Private Function cStep_IncVersionX() As String

    cStep_IncVersionX = mcStep.IncVersionX

End Function
Private Sub cStep_UpdateIteratorVersion()

    Call mcStep.UpdateIteratorVersion

End Sub

Private Function cStep_IteratorCount() As Long

    cStep_IteratorCount = mcStep.IteratorCount

End Function

Private Sub cStep_UnloadIterators()

    Call mcStep.UnloadIterators

End Sub

Private Sub cStep_DeleteIterator(cItRecord As cIterator)

    Call mcStep.DeleteIterator(cItRecord)

End Sub
Private Property Get cStep_IteratorName() As String

    cStep_IteratorName = mcStep.IteratorName

End Property
Private Property Let cStep_IteratorName(ByVal RHS As String)

    mcStep.IteratorName = RHS

End Property

Private Sub cStep_SaveIterators()

    Call mcStep.SaveIterators

End Sub
```

```
Private Sub cStep_LoadIterator(cItRecord As cIterator)

    Call mcStep.LoadIterator(cItRecord)

End Sub

Private Property Let cStep_Position(ByVal RHS As Long)

    mcStep.Position = RHS

End Property
Private Sub cStep_InsertIterator(cItRecord As cIterator)

    Call mcStep.InsertIterator(cItRecord)

End Sub
Private Function cStep_Iterators() As Variant

    cStep_Iterators = mcStep.Iterators

End Function
Private Sub cStep_ModifyIterator(cItRecord As cIterator)

    Call mcStep.ModifyIterator(cItRecord)

End Sub
Private Sub cStep_RemoveIterator(cItRecord As cIterator)

    Call mcStep.RemoveIterator(cItRecord)

End Sub
Private Sub cStep_UpdateIterator(cItRecord As cIterator)

    Call mcStep.UpdateIterator(cItRecord)

End Sub
Private Sub cStep_AddIterator(cItRecord As cIterator)

    Call mcStep.AddIterator(cItRecord)

End Sub

Private Property Get cStep_Position() As Long

    cStep_Position = mcStep.Position

End Property

Private Function cStep_Clone(Optional cCloneStep As cStep) As cStep

    Dim cNewManager As cManager

    Set cNewManager = New cManager
    Set cStep_Clone = mcStep.Clone(cNewManager)

End Function

Private Property Get cStep_IndOperation() As Operation

    cStep_IndOperation = mcStep.IndOperation

End Property

Private Property Let cStep_IndOperation(ByVal RHS As Operation)

    mcStep.IndOperation = RHS

End Property

Private Property Get cStep_NextStepId() As Long

    cStep_NextStepId = mcStep.NextStepId

End Property

Private Property Let cStep_OutputFile(ByVal RHS As String)

    mcStep.OutputFile = RHS

End Property

Private Property Get cStep_OutputFile() As String

    cStep_OutputFile = mcStep.OutputFile

End Property

Private Property Let cStep_ErrorFile(ByVal RHS As String)

    mcStep.ErrorFile = RHS

End Property

Private Property Get cStep_ErrorFile() As String

    cStep_ErrorFile = mcStep.ErrorFile

End Property
'Private Property Let cStep_LogFile(ByVal RHS As String)
'
'    mcStep.LogFile = RHS
'
'End Property
'
'Private Property Get cStep_LogFile() As String
'
'    cStep_LogFile = mcStep.LogFile
'
'End Property

Private Property Let cStep_ArchivedFlag(ByVal RHS As Boolean)
```

```
    mcStep.ArchivedFlag = RHS

End Property

Private Property Get cStep_ArchivedFlag() As Boolean

    cStep_ArchivedFlag = mcStep.ArchivedFlag

End Property

Private Property Get cStep_NodeDB() As DAO.Database

    Set cStep_NodeDB = mcStep.NodeDB

End Property

Private Sub Class_Initialize()

    ' Create the object
    Set mcStep = New cStep

    ' Initialize the object with valid values for a manager step
    ' The global flag should be the first field to be initialized
    ' since subsequent validations might try to check if the
    ' step being created is global
    mcStep.GlobalFlag = False
'    mcStep.GlobalRunMethod = gintNoOption
    mcStep.StepType = gintManagerStep

    ' Since the manager step does not take any action, the step
    ' text and file name will always be empty
    mcStep.StepText = gstrEmptyString
    mcStep.StepTextFile = gstrEmptyString

    ' Since the manager step does not take any action, execution
    ' properties for the step will be empty
    mcStep.ExecutionMechanism = gintNoOption
    mcStep.FailureDetails = gstrEmptyString
    mcStep.ContinuationCriteria = gintNoOption

End Sub
Private Sub Class_Terminate()

    ' Remove the step object
    Set mcStep = Nothing

End Sub
Private Sub cStep_Add()

    ' Call the Add method of the step class to carry out the insert
    mcStep.Add

End Sub
Private Property Get cStep_ContinuationCriteria() As
ContinuationCriteria

    cStep_ContinuationCriteria = mcStep.ContinuationCriteria

End Property

Private Property Let cStep_ContinuationCriteria(ByVal RHS As
ContinuationCriteria)

    ' Since a manager step cannot take any action, the continuation
    ' criteria property does not apply to it
    mcStep.ContinuationCriteria = gintNoOption

End Property

Private Property Let cStep_DegreeParallelism(ByVal RHS As String)

    mcStep.DegreeParallelism = RHS

End Property

Private Property Get cStep_DegreeParallelism() As String

    cStep_DegreeParallelism = mcStep.DegreeParallelism

End Property

Private Sub cStep_DeleteStep()

    On Error GoTo cStep_DeleteStepErr
    mstrSource = mstrModuleName & "cStep_DeleteStep"

    mcStep.Delete
    Exit Sub

cStep_DeleteStepErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "cStep_DeleteStep"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteStepFailed, _
            mstrSource, _
            LoadResString(errDeleteStepFailed)

End Sub

Private Property Get cStep_EnabledFlag() As Boolean

    cStep_EnabledFlag = mcStep.EnabledFlag

End Property

Private Property Let cStep_EnabledFlag(ByVal RHS As Boolean)

    mcStep.EnabledFlag = RHS

End Property
```

```
Private Property Let cStep_ExecutionMechanism(ByVal RHS As
ExecutionMethod)

    ' Since a manager step cannot take any action, the Execution
    ' Mechanism property does not apply to it
    mcStep.ExecutionMechanism = gintNoOption

End Property

Private Property Get cStep_ExecutionMechanism() As ExecutionMethod

    cStep_ExecutionMechanism = mcStep.ExecutionMechanism

End Property

Private Property Let cStep_FailureDetails(ByVal RHS As String)

    ' Since a manager step cannot take any action, the Failure
    ' Details property does not apply to it
    mcStep.FailureDetails = gstrEmptyString

End Property

Private Property Get cStep_FailureDetails() As String

    cStep_FailureDetails = mcStep.FailureDetails

End Property

Private Property Get cStep_GlobalFlag() As Boolean

    cStep_GlobalFlag = mcStep.GlobalFlag

End Property

Private Property Let cStep_GlobalFlag(ByVal RHS As Boolean)

    ' Set the global flag to false - this flag is initialized when
    ' an instance of the class is created. Just making sure that
    ' nobody changes the value inadvertently
    mcStep.GlobalFlag = False

End Property
Private Sub cStep_Modify()

    ' Call the Modify method of the step class to carry out the update
    mcStep.Modify

End Sub

Private Property Let cStep_ParentStepId(ByVal RHS As Long)

    mcStep.ParentStepId = RHS

End Property

Private Property Get cStep_ParentStepId() As Long

    cStep_ParentStepId = mcStep.ParentStepId

End Property

Private Property Let cStep_ParentVersionNo(ByVal RHS As String)

    mcStep.ParentVersionNo = RHS

End Property

Private Property Get cStep_ParentVersionNo() As String

    cStep_ParentVersionNo = mcStep.ParentVersionNo

End Property

Private Property Let cStep_SequenceNo(ByVal RHS As Integer)

    mcStep.SequenceNo = RHS

End Property

Private Property Get cStep_SequenceNo() As Integer

    cStep_SequenceNo = mcStep.SequenceNo

End Property

Private Property Let cStep_StepId(ByVal RHS As Long)

    mcStep.StepId = RHS

End Property

Private Property Get cStep_StepId() As Long

    cStep_StepId = mcStep.StepId

End Property


Private Property Let cStep_StepLabel(ByVal RHS As String)

    mcStep.StepLabel = RHS

End Property

Private Property Get cStep_StepLabel() As String

    cStep_StepLabel = mcStep.StepLabel

End Property
```

```
Private Property Let cStep_StepLevel(ByVal RHS As Integer)

    mcStep.StepLevel = RHS

End Property

Private Property Get cStep_StepLevel() As Integer

    cStep_StepLevel = mcStep.StepLevel

End Property

Private Property Let cStep_StepText(ByVal RHS As String)

    ' Since the manager step does not take any action, the step
    ' text and file name will always be empty
    mcStep.StepText = gstrEmptyString

End Property

Private Property Get cStep_StepText() As String

    cStep_StepText = mcStep.StepText

End Property

Private Property Let cStep_StepTextFile(ByVal RHS As String)

    ' Since the manager step does not take any action, the step
    ' text and file name will always be empty
    mcStep.StepTextFile = gstrEmptyString

End Property

Private Property Get cStep_StepTextFile() As String

    cStep_StepTextFile = mcStep.StepTextFile

End Property

Private Property Let cStep_StepType(RHS As gintStepType)

    mcStep.StepType = gintManagerStep

End Property

Private Property Get cStep_StepType() As gintStepType

    cStep_StepType = mcStep.StepType

End Property

Private Sub cStep_Validate()
    ' The validate routines for each of the steps will
    ' carry out the specific validations for the type and
    ' call the generic validation routine
    On Error GoTo cStep_ValidateErr
    mstrSource = mstrModuleName & "cStep_Validate"

    ' Validations specific to manager steps

    ' Check if the step text or a file name has been
    ' specified
    If Not StringEmpty(mcStep.StepText) Or Not
StringEmpty(mcStep.StepTextFile) Then
        ShowError errTextAndFileNullForManager
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.ExecutionMechanism <> gintNoOption Then
        ShowError errExecutionMechanismInvalid
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.FailureDetails <> gstrEmptyString Then
        ShowError errFailureDetailsNullForMgr
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.ContinuationCriteria <> gintNoOption Then
        ShowError errContCriteriaInvalid
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    mcStep.Validate

    Exit Sub

cStep_ValidateErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "cStep_Validate"
    On Error GoTo 0
    Err.Raise vbObjectError + errValidateFailed, _
            mstrSource, _
            LoadResString(errValidateFailed)
End Sub

Private Property Let cStep_VersionNo(ByVal RHS As String)
```

```
    mcStep.VersionNo = RHS

End Property

Private Property Get cStep_VersionNo() As String

    cStep_VersionNo = mcStep.VersionNo

End Property

Private Property Let cStep_WorkspaceId(ByVal RHS As Long)

    mcStep.WorkspaceId = RHS

End Property

Private Property Get cStep_WorkspaceId() As Long

    cStep_WorkspaceId = mcStep.WorkspaceId

End Property
```

cNode.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cNode"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cNode.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Defines the properties that an object has to implement.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Property Get IndOperation() As Operation
End Property
Public Property Let IndOperation(ByVal vdata As Operation)
End Property
Public Sub Validate()
End Sub
Public Property Get Value() As String
End Property
Public Property Let Value(ByVal vdata As String)
End Property

Public Property Get NodeDB() As Database
End Property
Public Property Set NodeDB(vdata As Database)
End Property

Public Property Get Position() As Long
End Property
Public Property Let Position(ByVal vdata As Long)
End Property
```

cNodeCollections.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cNodeCollections"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cNodeCollections.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Implements an array of objects.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Node counter
Private mlngNodeCount As Long
Private mdbsNodeDb As Database
Private mcarrNodes() As Object

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cNodeCollections."

Public Property Set Item(ByVal Position As Long, _
        ByVal objNode As Object)

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mlngNodeCount Then
        Set mcarrNodes(Position) = objNode
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Get Item(ByVal Position As Long) As Object
Attribute Item.VB_UserMemId = 0
```

```
    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mlngNodeCount Then
        Set Item = mcarrNodes(Position)
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property

Public Sub Commit(ByVal cSaveObj As Object, _
        ByVal lngIndex As Long)
    ' This procedure checks if any changes have been made to the
    ' passed in object. If so, it calls the corresponding method
    ' to commit the changes.

    On Error GoTo CommitErr
    mstrSource = mstrModuleName & "Commit"

    Select Case cSaveObj.IndOperation
        Case QueryOp
            ' No changes were made to the queried parameter.
            ' Do nothing

        Case InsertOp
            cSaveObj.Add
            cSaveObj.IndOperation = QueryOp

        Case UpdateOp
            cSaveObj.Modify
            cSaveObj.IndOperation = QueryOp

        Case DeleteOp
            cSaveObj.Delete
            ' Now we can remove the record from the array
            Call Unload(lngIndex)

    End Select

    Exit Sub

CommitErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Commit"
    On Error GoTo 0
    Err.Raise vbObjectError + errCommitFailed, _
            mstrSource, _
            LoadResString(errCommitFailed)

End Sub
Public Sub Save(ByVal lngWorkspace As Long)
    ' Calls a procedure to commit all changes for the passed
    ' in workspace.

    Dim lngIndex As Long

    On Error GoTo SaveErr

    ' Find all parameters in the array with a matching workspace id
    ' It is important to step backwards through the array, since
    ' we delete parameter records as we go along!
    For lngIndex = mlngNodeCount - 1 To 0 Step -1
        If mcarrNodes(lngIndex).WorkspaceId = lngWorkspace Then

            ' Call a procedure to commit all changes to the
            ' parameter record, if any
            Call Commit(mcarrNodes(lngIndex), lngIndex)

        End If
    Next lngIndex

    Exit Sub

SaveErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Save"
    On Error GoTo 0
    Err.Raise vbObjectError + errSaveFailed, _
            mstrSource, _
            LoadResString(errSaveFailed)

End Sub
Public Property Get Count() As Long

    Count = mlngNodeCount

End Property

Public Property Get NodeDB() As Database

    Set NodeDB = mdbsNodeDb

End Property
Public Property Set NodeDB(vdata As Database)

    Set mdbsNodeDb = vdata

End Property

Public Sub Load(cNodeToLoad As Object)
    ' Adds the passed in object to the array

    On Error GoTo LoadErr

    ' If this procedure is called by the add to array procedure,
    ' the database object has already been initialized
    If cNodeToLoad.NodeDB Is Nothing Then

        ' All the Nodes will be initialized with the database
        ' objects before being added to the array
```

```
        Set cNodeToLoad.NodeDB = mdbsNodeDb

    End If

    ReDim Preserve mcarrNodes(mlngNodeCount)

    ' Set the newly added element in the array to the passed in Node
    cNodeToLoad.Position = mlngNodeCount
    Set mcarrNodes(mlngNodeCount) = cNodeToLoad

    mlngNodeCount = mlngNodeCount + 1

    Exit Sub

LoadErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadFailed, mstrModuleName & "Load", _
            LoadResString(errLoadFailed)

End Sub
Public Sub Unload(lngDeletePosition As Long)
    ' Unloads the passed in object from the array

    On Error GoTo UnloadErr

    If lngDeletePosition < (mlngNodeCount - 1) Then

        ' Set the Node at the position being deleted to
        ' the last Node in the Node array
        Set mcarrNodes(lngDeletePosition) = mcarrNodes(mlngNodeCount -
1)
        mcarrNodes(lngDeletePosition).Position = lngDeletePosition
    End If

    ' Delete the last Node from the array
    mlngNodeCount = mlngNodeCount - 1
    If mlngNodeCount > 0 Then
        ReDim Preserve mcarrNodes(0 To mlngNodeCount - 1)
    Else
        ReDim mcarrNodes(0)
    End If

    Exit Sub

UnloadErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Unload"
    On Error GoTo 0
    Err.Raise vbObjectError + errUnloadFailed, _
            mstrSource, _
            LoadResString(errUnloadFailed)

End Sub
Public Sub Delete(lngDeletePosition As Long)
    ' Deletes the object at the specified position in the
    ' array

    Dim cDeleteObj As Object

    On Error GoTo DeleteErr
    mstrSource = mstrModuleName & "Delete"

    Set cDeleteObj = mcarrNodes(lngDeletePosition)

    If cDeleteObj.IndOperation = InsertOp Then
        ' If we are deleting a record that has just been inserted,
        ' blow it away
        Call Unload(lngDeletePosition)
    Else
        ' Set the operation for the deleted object to indicate a
        ' delete - we actually delete the element only at the time
        ' of a save operation
        cDeleteObj.IndOperation = DeleteOp
    End If

    Exit Sub

DeleteErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteFailed, _
            mstrSource, _
            LoadResString(errDeleteFailed)

End Sub
Public Sub Modify(cModifiedNode As Object)
    ' Sets the object at the passed in position to the
    ' modified object passed in

    On Error GoTo ModifyErr

    ' First check if the record is valid - all objects that
    ' use this collection class must have a Validate routine
    cModifiedNode.Validate

    ' If we are updating a record that hasn't yet been inserted,
    ' do not change the operation indicator - or we try to update
    ' a non-existant record
    If cModifiedNode.IndOperation <> InsertOp Then
        ' Set the operations to indicate an update
        cModifiedNode.IndOperation = UpdateOp
    End If

    ' Modify the object at the queried position - the Position
    ' will be maintained by this class
    Set mcarrNodes(cModifiedNode.Position) = cModifiedNode

    Exit Sub

ModifyErr:
```

```
    LogErrors Errors
    mstrSource = mstrModuleName & "Modify"
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyFailed, _
            mstrSource, _
            LoadResString(errModifyFailed)

End Sub
Public Sub Add(cNodeToAdd As Object)

    On Error GoTo AddErr

    Set cNodeToAdd.NodeDB = mdbsNodeDb

    ' First check if the record is valid
    cNodeToAdd.Validate

    ' Set the operation to indicate an insert
    cNodeToAdd.IndOperation = InsertOp

    ' Call a procedure to load the record in the array
    Call Load(cNodeToAdd)

    Exit Sub

AddErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Add"
    On Error GoTo 0
    Err.Raise vbObjectError + errAddFailed, _
            mstrSource, _
            LoadResString(errAddFailed)

End Sub

Private Sub Class_Terminate()

    ReDim mcarrNodes(0)
    mlngNodeCount = 0

End Sub
```

Common.bas

```
Attribute VB_Name = "Common"
'  FILE:       Common.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Module containing common functionality throughout
'              StepMaster
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private Const mstrModuleName As String = "Common."

' Used to separate the variable data from the constant error
' message being raised when a context-sensitive error is displayed
Private Const mintDelimiter As String = " : "
Private Const mstrFormatString = "mmddyy"

' Identifiers for the different labels that need to be loaded
' into the tree view for each workspace
Public Const mstrWorkspacePrefix = "W"
Public Const mstrParameterPrefix = "P"
Public Const mstrParamConnectionPrefix = "C"
Public Const mstrConnectionDtlPrefix = "N"
Public Const mstrParamExtensionPrefix = "E"
Public Const mstrParamBuiltInPrefix = "B"
Public Const gstrGlobalStepPrefix = "G"
Public Const gstrManagerStepPrefix = "M"
Public Const gstrWorkerStepPrefix = "S"
Public Const gstrDummyPrefix = "D"
Public Const mstrLabelPrefix = "L"
Public Const mstrInstancePrefix = "I"
Public Function LabelStep(lngWorkspaceIdentifier As Long) As String
    ' Returns the step label for the workspace identifier passed in
    ' Basically this is a wrapper around the MakeKeyValid function

    LabelStep = MakeKeyValid(gintStepLabel, gintStepLabel,
lngWorkspaceIdentifier)

End Function


Public Function JulianDateToString(dt64Bit As Currency) As String

    Dim lYear As Long
    Dim lMonth As Long
    Dim lDay As Long
    Dim lHour As Long
    Dim lMin As Long
    Dim lSec As Long
    Dim lMs As Long

    Call JulianToTime(dt64Bit, lYear, lMonth, lDay, lHour, lMin, lSec,
lMs)
    JulianDateToString = Format$(lYear, gsYearFormat) & gsDateSeparator
& _
            Format$(lMonth, gsDtFormat) & gsDateSeparator & _
            Format$(lDay, gsDtFormat) & gstrBlank & _
            Format$(lHour, gsTmFormat) & gsTimeSeparator & _
            Format$(lMin, gsTmFormat) & gsTimeSeparator & _
            Format$(lSec, gsTmFormat) & gsMsSeparator & _
            Format$(lMs, gsMSecondFormat)

End Function
```

```
Public Sub DeleteFile(strFile As String, Optional ByVal bCheckIfEmpty As
Boolean = False)

    ' Ensure that there is only a single file of the name before delete,
since
    ' Kill supports wildcards and can potentially delete a number of
files
    Dim strTemp As String

    If CheckFileExists(strFile) Then
        If bCheckIfEmpty Then
            If FileLen(strFile) = 0 Then
                Kill strFile
            End If
        Else
            Kill strFile
        End If
    End If

End Sub
Public Function CheckFileExists(strFile As String) As Boolean

    ' Returns true if the passed in file exists
    ' Raises an error if multiple files are found (filename contains a
wildcard)
    CheckFileExists = False

    If Not StringEmpty(Dir(strFile)) Then
        If Not StringEmpty(Dir()) Then
            On Error GoTo 0
            Err.Raise vbObjectError + errDeleteSingleFile, _
                mstrModuleName & "DeleteFile",
LoadResString(errDeleteSingleFile)
        End If

        CheckFileExists = True
    End If

End Function

Public Function GetVersionString() As String
GetVersionString = "Version " & gsVersion
End Function

Function IsLabel(strKey As String) As Boolean

    ' The tree view control on frmMain can contain two types of
    ' nodes -
    ' 1. Nodes that contain data for the workspace - this could
    ' be data for the different types of steps or parameters
    ' 2. Nodes that display static data - these kind of nodes
    ' are referred to as label nodes e.g. "Global Steps" is a
    ' label node
    ' This function returns True if the passed in key corresponds
    ' to a label node

    IsLabel = InStr(strKey, mstrLabelPrefix) > 0

End Function

Function MakeKeyValid(lngIdentifier As Long, _
    intTypeOfNode As Integer, _
    Optional ByVal WorkspaceId As Long = 0, _
    Optional ByVal InstanceId As Long = 0) As String

    ' We use a numbering scheme while loading the tree view with
    ' all node data, since it needs a unique key and we want to
    ' use the key to identify the data it contains.
    ' Moreover, add a character to the beginning of the identifier
    ' so that the tree view control accepts it as a valid string,
    ' viz. "456" doesn't work, so change it to "W456"
    ' The general scheme is to concatenate a Label with the Identifier
    ' e.g A Global Step Node will have the Label, G and the Step Id
    ' concatenated to form the unique key
    ' The list of all such node types is given below
    ' 1. "W" + Workspace_Id for Workspace nodes
    ' 2. "P" + Parameter_Id for Parameter nodes
    ' 3. "M" + Step_Id for Manager Step nodes
    ' 4. "S" + Step_Id for Worker Step nodes
    ' 5. "G" + Step_Id for Global Step nodes
    ' 6. Instance_id + "I" + Step_Id for Instance nodes
    ' 7. Workspace_id + "L" + the label identifier = node type for all
Label nodes
    ' Since the manager, worker and global steps are stored in the
    ' same table and the step identifiers will always be unique, we
    ' can use the same character as the prefix, but this is a
    ' convenient way to know the type of step being processed.
    ' The workspace id is appended to the label identifier to make
    ' it unique, since multiple workspaces may be open during a session
    ' Strip the prefix characters off while saving the Ids to the db

    Dim strPrefixChar As String

    On Error GoTo MakeKeyValidErr
    gstrSource = mstrModuleName & "MakeKeyValid"

    Select Case intTypeOfNode
        Case gintWorkspace
            strPrefixChar = mstrWorkspacePrefix
        Case gintGlobalStep
            strPrefixChar = gstrGlobalStepPrefix
        Case gintManagerStep
            strPrefixChar = gstrManagerStepPrefix
        Case gintWorkerStep
            strPrefixChar = gstrWorkerStepPrefix
        Case gintRunManager, gintRunWorker
            If InstanceId = 0 Then
                On Error GoTo 0
                Err.Raise vbObjectError + errMandatoryParameterMissing,
_
                    gstrSource, _
                    LoadResString(errMandatoryParameterMissing)
```

```
            End If
            ' Concatenate the instance identifier and the step
            ' identifier to form a unique key
            strPrefixChar = Trim$(Str$(InstanceId)) & mstrInstancePrefix
        Case gintParameter
            strPrefixChar = mstrParameterPrefix
        Case gintNodeParamConnection
            strPrefixChar = mstrParamConnectionPrefix
        Case gintConnectionDtl
            strPrefixChar = mstrConnectionDtlPrefix
        Case gintNodeParamExtension
            strPrefixChar = mstrParamExtensionPrefix
        Case gintNodeParamBuiltIn
            strPrefixChar = mstrParamBuiltInPrefix
        Case gintGlobalsLabel, gintParameterLabel,
gintParamConnectionLabel, _
                gintConnDtlLabel, _
                gintParamExtensionLabel, gintParamBuiltInLabel,
gintGlobalStepLabel, _
                gintStepLabel
            If WorkspaceId = 0 Then
                ' The Workspace Id has to be specified for a label node
                ' Otherwise it will not be possible to generate unique
label
                ' identifiers if multiple workspaces are open
                On Error GoTo 0
                Err.Raise vbObjectError + errWorkspaceIdMandatory, _
                    gstrSource, _
                    LoadResString(errWorkspaceIdMandatory)
            End If
            ' For all labels, the workspace identifier and the
            ' label prefix are concatenated to form the key
            strPrefixChar = Trim$(Str$(WorkspaceId)) & mstrLabelPrefix
        Case Else
                On Error GoTo 0
                Err.Raise vbObjectError + errInvalidNodeType, _
                    gstrSource, _
                    LoadResString(errInvalidNodeType)
    End Select

    MakeKeyValid = strPrefixChar & Trim$(Str$(lngIdentifier))

    Exit Function

MakeKeyValidErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errMakeKeyValidFailed, _
            gstrSource, _
            LoadResString(errMakeKeyValidFailed)

End Function

Function MakeIdentifierValid(strKey As String) As Long

    ' Returns the Identifier corresponding to the passed in key
    ' (Reverse of what was done in MakeKeyValid)

    On Error GoTo MakeIdentifierValidErr

    If IsLabel(strKey) Then
        ' If the key corresponds to a label node, the identifier
        ' appears to the right of the label prefix
        MakeIdentifierValid = Val(Mid(strKey, InStr(strKey,
mstrLabelPrefix) + 1))
    ElseIf InStr(strKey, mstrInstancePrefix) = 0 Then
        ' For all other nodes, stripping the first character off
        ' returns a valid Id
        MakeIdentifierValid = Val(Mid(strKey, 2))
    Else
        ' Instance node - strip of all characters till the
        ' instance prefix
        MakeIdentifierValid = Val(Mid(strKey, InStr(strKey,
mstrInstancePrefix) + 1))
    End If

    Exit Function

MakeIdentifierValidErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errMakeIdentifierValidFailed, _
            mstrModuleName & "MakeIdentifierValid", _
            LoadResString(errMakeIdentifierValidFailed)

End Function
Public Function IsInstanceNode(strNodeKey As String) As Boolean

    ' Returns true if the passed in node key corresponds to a step
instance
    IsInstanceNode = InStr(strNodeKey, mstrInstancePrefix) > 0

End Function
Public Function IsBuiltInLabel(strNodeKey As String) As Boolean

    ' Returns true if the passed in node key corresponds to a step
instance
    IsBuiltInLabel = (IsLabel(strNodeKey) And _
        (MakeIdentifierValid(strNodeKey) = gintParamBuiltInLabel))

End Function

Public Sub ShowBusy()
    ' Modifies the mousepointer to indicate that the
    ' application is busy

    On Error Resume Next

    Screen.MousePointer = vbHourglass

End Sub
Public Sub ShowFree()
```

```
    ' Modifies the mousepointer to indicate that the
    ' application has finished processing and is ready
    ' to accept user input

    On Error Resume Next

    Screen.MousePointer = vbDefault

End Sub

Public Function InstrR(strMain As String, _
        strSearch As String) As Integer
    ' Finds the last occurrence of the passed in string

    Dim intPos As Integer
    Dim intPrev As Integer

    On Error GoTo InstrRErr

    intPrev = intPos
    intPos = InStr(1, strMain, strSearch)

    Do While intPos > 0
        intPrev = intPos
        intPos = InStr(intPos + 1, strMain, strSearch)
    Loop
    InstrR = intPrev

    Exit Function

InstrRErr:
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "InstrR"
    On Error GoTo 0
    Err.Raise vbObjectError + errInstrRFailed, _
        gstrSource, _
        LoadResString(errInstrRFailed)

End Function

Public Function GetDefaultDir(lWspId As Long, WspParameters As
cArrParameters) As String

    Dim sDir As String
    sDir = SubstituteParameters( _
            gstrEnvVarSeparator & PARAM_DEFAULT_DIR &
gstrEnvVarSeparator, _
            lWspId, WspParameters:=WspParameters)
    MakePathValid (sDir & gstrFileSeparator & "a.txt")
    GetDefaultDir = GetShortName(sDir)
    If StringEmpty(GetDefaultDir) Then
        GetDefaultDir = App.Path
    End If

End Function

Public Sub AddArrayElement(ByRef arrNodes() As Object, _
        ByVal objToAdd As Object, _
        ByRef lngCount As Long)
    ' Adds the passed in object to the array

    On Error GoTo AddArrayElementErr

    ' Increase the array dimension and add the object to it
    ReDim Preserve arrNodes(lngCount)
    Set arrNodes(lngCount) = objToAdd
    lngCount = lngCount + 1

    Exit Sub

AddArrayElementErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "AddArrayElement"
    On Error GoTo 0
    Err.Raise vbObjectError + errAddArrayElementFailed, _
            gstrSource, _
            LoadResString(errAddArrayElementFailed)

End Sub


Public Function CheckForNullField(rstRecords As Recordset, strFieldName
As String) As String

    ' Returns an empty string if a given field is null
    On Error GoTo CheckForNullFieldErr

    If IsNull(rstRecords.Fields(strFieldName)) Then
        CheckForNullField = gstrEmptyString
    Else
        CheckForNullField = rstRecords.Fields(strFieldName)
    End If
    Exit Function

CheckForNullFieldErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errCheckForNullFieldFailed, _
            mstrModuleName & "CheckForNullField", _
            LoadResString(errCheckForNullFieldFailed)

End Function

Public Function ErrorOnNullField(rstRecords As Recordset, strFieldName
As String) As Variant

    ' If a given field is null, raises an error
    ' Else, returns the field value in a variant
    ' The calling function must convert the return value to the
    ' appropriate type
    On Error GoTo ErrorOnNullFieldErr
    gstrSource = mstrModuleName & "ErrorOnNullField"
```

```
    If IsNull(rstRecords.Fields(strFieldName)) Then
        On Error GoTo 0
        Err.Raise vbObjectError + errMandatoryFieldNull, _
            gstrSource, _
            strFieldName & mintDelimiter &
LoadResString(errMandatoryFieldNull)
    Else
        ErrorOnNullField = rstRecords.Fields(strFieldName)
    End If
    Exit Function

ErrorOnNullFieldErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errUnableToCheckNull, _
        gstrSource, _
        strFieldName & mintDelimiter &
LoadResString(errUnableToCheckNull)

End Function
Public Function StringEmpty(strCheckString As String) As Boolean

    StringEmpty = (strCheckString = gstrEmptyString)

End Function
Public Function GetIteratorValue(cStepIterators As cRunColIt, _
        ByVal strItName As String)

    Dim lngIndex As Long
    Dim strValue As String

    On Error GoTo GetIteratorValueErr
    gstrSource = mstrModuleName & "GetIteratorValue"

    ' Find the iterator in the Iterators collection
    For lngIndex = 0 To cStepIterators.Count - 1
        If cStepIterators(lngIndex).IteratorName = strItName Then
            strValue = cStepIterators(lngIndex).Value
            Exit For
        End If
    Next lngIndex

    If lngIndex > cStepIterators.Count - 1 Then
        ' The iterator has not been defined for the branch
        ' Raise an error
        On Error GoTo 0
        Err.Raise vbObjectError + errParamNameInvalid, _
                gstrSource, _
                LoadResString(errParamNameInvalid)
    End If

    GetIteratorValue = strValue
    Exit Function

GetIteratorValueErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "GetIteratorValue"
    On Error GoTo 0
    Err.Raise vbObjectError + errGetParamValueFailed, _
            gstrSource, _
            LoadResString(errGetParamValueFailed)

End Function
Public Function SubstituteParameters(ByVal strComString As String, _
        ByVal lngWorkspaceId As Long, _
        Optional StepIterators As cRunColIt = Nothing, _
        Optional WspParameters As cArrParameters = Nothing) As String
    ' This function substitutes all parameter names and
    ' environment variables in the passed in string with
    ' their values. It also substitutes the value for the
    ' iterators, if any.
    ' Since the syntax is to enclose parameter names and
    ' environment variables in "%", we check if a given
    ' variable is a parameter - if so, we substitute the
    ' parameter value - else we try to get the value from
    ' the environment

    Dim intPos As Integer
    Dim intEndPos As Integer
    Dim strEnvVariable As String
    Dim strValue As String
    Dim strCommand As String
    Dim cTempStr As cStringSM

    ' Initialize the return value of the function to the
    ' passed in command
    strCommand = strComString

    If WspParameters Is Nothing Then Set WspParameters = gcParameters

    Set cTempStr = New cStringSM

    intPos = InStr(strCommand, gstrEnvVarSeparator)
    Do While intPos <> 0
        If Mid(strCommand, intPos + 1, 1) = gstrEnvVarSeparator Then
            ' Wildcard character - to be substituted by a single % -
later!
            intPos = intPos + 2
            If intPos > Len(strCommand) Then Exit Do
        Else
            ' Extract the environment variable from the passed
            ' in string
            intEndPos = InStr(intPos + 1, strCommand,
gstrEnvVarSeparator)

            If intEndPos > 0 Then
                strEnvVariable = Mid(strCommand, intPos + 1, intEndPos -
intPos - 1)
            Else
                On Error GoTo 0
```

```
                Err.Raise vbObjectError + errParamSeparatorMissing, _
                        gstrSource, _
                        LoadResString(errParamSeparatorMissing)
            End If
            strValue = gstrEmptyString

            ' Get the value of the variable and call a function
            ' to replace the variable with it's value
            strValue = GetValue(strEnvVariable, lngWorkspaceId,
StepIterators, WspParameters)
            ' The function raises an error if the variable is
            ' not found
            strCommand = cTempStr.ReplaceSubString(strCommand, _
                    gstrEnvVarSeparator & strEnvVariable &
gstrEnvVarSeparator, _
                    strValue)
        End If

        intPos = InStr(intPos, strCommand, gstrEnvVarSeparator)
    Loop

    strCommand = cTempStr.ReplaceSubString(strCommand, _
            gstrEnvVarSeparator & gstrEnvVarSeparator,
gstrEnvVarSeparator)

    Set cTempStr = Nothing
    SubstituteParameters = strCommand

End Function
Private Function GetValue(ByVal strParameter As String, _
        ByVal lngWorkspaceId As Long, _
        cStepIterators As cRunColIt, _
        WspParameters As cArrParameters) As String
    ' This function returns the value for the passed in
    ' parameter - it may be a workspace parameter, an
    ' environment variable or an iterator

    Dim intPos As Integer
    Dim intEndPos As Integer
    Dim strVariable As String
    Dim strValue As String
    Dim cParamRec As cParameter

    On Error GoTo GetValueErr

    ' Initialize the return value of the function to the
    ' empty
    strValue = gstrEmptyString

    intPos = InStr(strParameter, gstrEnvVarSeparator)
    If intPos > 0 Then
        ' Extract the variable from the passed in string
        intEndPos = InStr(intPos + 1, strParameter, gstrEnvVarSeparator)
        If intEndPos = 0 Then
            intEndPos = Len(strParameter)
        End If
        strVariable = Mid(strParameter, intPos + 1, intEndPos - intPos -
1)
    Else
        ' The separator charactor has not been passed in -
        ' try to find the value of the passed in parameter
        strVariable = strParameter
    End If

    If Not StringEmpty(strVariable) Then
        ' Check if this is the timestamp parameter first
        If strVariable = gstrTimeStamp Then
            strValue = Format$(Now, mstrFormatString, _
                    vbUseSystemDayOfWeek, vbUseSystem)
        Else
            ' Try to find a parameter for the workspace with
            ' the same name
            Set cParamRec =
WspParameters.GetParameterValue(lngWorkspaceId, _
                    strVariable)
            If cParamRec Is Nothing Then
                If Not cStepIterators Is Nothing Then
                    ' If the string is not a parameter, then check
                    ' if it is an iterator
                    strValue = GetIteratorValue(cStepIterators,
strVariable)
                End If

                If StringEmpty(strValue) Then
                    ' Neither - Check if it is an environment variable
                    strValue = Environ$(strVariable)
                    If StringEmpty(strValue) Then
                        On Error GoTo 0
                        WriteError errSubValuesFailed, _
                                OptArgs:="Invalid parameter: " & gstrSQ
& strVariable & gstrSQ
                        Err.Raise vbObjectError + errSubValuesFailed, _
                                mstrModuleName & "GetValue", _
                                LoadResString(errSubValuesFailed) &
"Invalid parameter: " & gstrSQ & strVariable & gstrSQ
                    End If
                End If
            Else
                strValue = cParamRec.ParameterValue
            End If
        End If
    End If

    GetValue = strValue

    Exit Function

GetValueErr:
    If Err.Number = vbObjectError + errParamNameInvalid Then
        ' If the parameter has not been defined for the
        ' workspace then check if it is an environment
```

```
        ' variable
        Resume Next
    End If

    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "GetValue"
    WriteError errSubValuesFailed, gstrSource, "Parameter: " & gstrSQ &
strVariable & gstrSQ
    On Error GoTo 0
    Err.Raise vbObjectError + errSubValuesFailed, _
            gstrSource, _
            LoadResString(errSubValuesFailed) & "Parameter: " & gstrSQ &
strVariable & gstrSQ

End Function
Public Function SQLFixup(strField As String) As String
    ' Returns a string that can be executed by SQL Server

    Dim cMyStr As New cStringSM
    Dim strTemp As String

    On Error GoTo SQLFixupErr

    strTemp = strField
    SQLFixup = strTemp

    ' Single-quotes have to be replaced by two single-quotes,
    ' since a single-quote is the identifier delimiter
    ' character - call a procedure to do the replace
'    SQLFixup = cMyStr.ReplaceSubString(strTemp, gstrDQ, "\" & gstrDQ)

    ' Replace pipe characters with the corresponding chr function
    ' SQLFixup = cMyStr.ReplaceSubString(strTemp, gstrDQ, gstrDQ &
gstrDQ)

    Exit Function

SQLFixupErr:
    gstrSource = mstrModuleName & "SQLFixup"
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errMakeFieldValidFailed, _
            gstrSource, LoadResString(errMakeFieldValidFailed)

End Function
Public Function TranslateStepLabel(sLabel As String) As String
    ' Translates the passed in step label to a valid file name
    ' All characters in the label that are invalid for filenames (viz.
\ / : * ? " < > |)
    ' and spaces are substituted with underscores - also ensure that the
resulting filename
    ' is not greater than 255 characters
    Dim cTempStr As New cStringSM
    TranslateStepLabel = cTempStr.ReplaceSubString(sLabel,
gstrFileSeparator, "_")
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
"/", gstrUnderscore)
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
":", gstrUnderscore)
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
"*", gstrUnderscore)
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
"?", gstrUnderscore)
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
gstrDQ, gstrUnderscore)
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
"<", gstrUnderscore)
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
">", gstrUnderscore)
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
"|", gstrUnderscore)
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
gstrBlank, gstrUnderscore)

    ' Commas are substituted with underscores since the command shell
uses a comma to
    ' delimit commands
    TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
",", gstrUnderscore)

    If Len(TranslateStepLabel) > MAX_PATH Then
        TranslateStepLabel = Mid(TranslateStepLabel, 1, MAX_PATH)
    End If

End Function

Public Function TypeOfObject(ByVal objNode As Object) As Integer
    ' Determines the type of object that is passed in

    On Error GoTo TypeOfObjectErr
    gstrSource = mstrModuleName & "TypeOfObject"

    Select Case TypeName(objNode)
        Case "cWorkspace"
            TypeOfObject = gintWorkspace

        Case "cParameter"
            TypeOfObject = gintParameter

        Case "cConnection"
            TypeOfObject = gintParameterConnect

        Case "cConnDtl"
            TypeOfObject = gintConnectionDtl

        Case "cGlobalStep"
            TypeOfObject = gintGlobalStep

        Case "cManager"
            TypeOfObject = gintManagerStep
```

```
        Case "cWorker"
            TypeOfObject = gintWorkerStep

        Case "cStep"
            ' If a step record is passed in, call a function
            ' to determine the type of step
            TypeOfObject = TypeOfStep(StepClass:=objNode)

        Case Else
            WriteError errTypeOfObjectFailed, gstrSource, _
                    TypeName(objNode)
            On Error GoTo 0
            Err.Raise vbObjectError + errTypeOfObjectFailed, _
                    gstrSource, _
                    LoadResString(errTypeOfObjectFailed)
    End Select

    Exit Function

TypeOfObjectErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errTypeOfObjectFailed, _
            gstrSource, _
            LoadResString(errTypeOfObjectFailed)

End Function
```

ConnDtlCommon.bas

```
Attribute VB_Name = "ConnDtlCommon"
'  FILE:       ConnDtlCommon.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains functionality common across StepMaster and
'              SMRunOnly, pertaining to connections
'              Specifically, functions to load connections in an array
'              and so on.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "ConnDtlCommon."

Public Sub LoadRSInConnDtlArray(rstConns As Recordset, cConns As
cConnDtls)

    Dim cNewConnDtl As cConnDtl

    On Error GoTo LoadRSInConnDtlArrayErr

    If rstConns.RecordCount = 0 Then
        Exit Sub
    End If

    rstConns.MoveFirst
    While Not rstConns.EOF

        Set cNewConnDtl = New cConnDtl

        ' Initialize ConnDtl values
        ' Call a procedure to raise an error if mandatory fields are
null.
        cNewConnDtl.ConnNameId = ErrorOnNullField(rstConns,
FLD_ID_CONN_NAME)
        cNewConnDtl.WorkspaceId = ErrorOnNullField(rstConns,
FLD_ID_WORKSPACE)
        cNewConnDtl.ConnName = CStr(ErrorOnNullField(rstConns,
FLD_CONN_DTL_CONNECTION_NAME))
        cNewConnDtl.ConnectionString = CheckForNullField(rstConns,
FLD_CONN_DTL_CONNECTION_STRING)
        cNewConnDtl.ConnType = CheckForNullField(rstConns,
FLD_CONN_DTL_CONNECTION_TYPE)

        cConns.Load cNewConnDtl

        Set cNewConnDtl = Nothing
        rstConns.MoveNext
    Wend

    Exit Sub

LoadRSInConnDtlArrayErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadRSInConnDtlArray"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, gstrSource, _
            LoadResString(errLoadRsInArrayFailed)
End Sub
```

ConnectionCommon.bas

```
Attribute VB_Name = "ConnectionCommon"
'  FILE:       ConnnectionCommon.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains functionality common across StepMaster and
'              SMRunOnly, pertaining to connection strings
'              Specifically, functions to load connections strings
'              in an array and so on.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
```

```
' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "ConnectionCommon."

Public Sub LoadRecordsetInConnectionArray(rstConns As Recordset, cConns
As cConnections)

    Dim cNewConnection As cConnection

    On Error GoTo LoadRecordsetInConnectionArrayErr

    If rstConns.RecordCount = 0 Then
        Exit Sub
    End If

    rstConns.MoveFirst
    While Not rstConns.EOF

        Set cNewConnection = New cConnection

        ' Initialize Connection values
        ' Call a procedure to raise an error if mandatory fields are
null.
        cNewConnection.ConnectionId = ErrorOnNullField(rstConns,
"connection_id")
        cNewConnection.WorkspaceId = CStr(ErrorOnNullField(rstConns,
FLD_ID_WORKSPACE))
        cNewConnection.ConnectionName = CStr(ErrorOnNullField(rstConns,
"connection_name"))
        cNewConnection.ConnectionValue = CheckForNullField(rstConns,
"connection_value")
        cNewConnection.Description = CheckForNullField(rstConns,
"description")

        cNewConnection.NoCountDisplay = CheckForNullField(rstConns,
"no_count_display")
        cNewConnection.NoExecute = CheckForNullField(rstConns,
"no_execute")
        cNewConnection.ParseQueryOnly = CheckForNullField(rstConns,
"parse_query_only")
        cNewConnection.QuotedIdentifiers = CheckForNullField(rstConns,
"ANSI_quoted_identifiers")
        cNewConnection.AnsiNulls = CheckForNullField(rstConns,
"ANSI_nulls")
        cNewConnection.ShowQueryPlan = CheckForNullField(rstConns,
"show_query_plan")
        cNewConnection.ShowStatsTime = CheckForNullField(rstConns,
"show_stats_time")
        cNewConnection.ShowStatsIO = CheckForNullField(rstConns,
"show_stats_io")
        cNewConnection.ParseOdbcMsg = CheckForNullField(rstConns,
"parse_odbc_msg_prefixes")
        cNewConnection.RowCount = CheckForNullField(rstConns,
"row_count")
        cNewConnection.TsqlBatchSeparator = CheckForNullField(rstConns,
"tsql_batch_separator")
        cNewConnection.QueryTimeOut = CheckForNullField(rstConns,
"query_time_out")
        cNewConnection.ServerLanguage = CheckForNullField(rstConns,
"server_language")
        cNewConnection.CharacterTranslation  = CheckForNullField(rstConns,
"character_translation")
        cNewConnection.RegionalSettings = CheckForNullField(rstConns,
"regional_settings")

        cConns.Load cNewConnection

        Set cNewConnection = Nothing
        rstConns.MoveNext
    Wend

    Exit Sub

LoadRecordsetInConnectionArrayErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadRecordsetInConnectionArray"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, gstrSource, _
            LoadResString(errLoadRsInArrayFailed)
End Sub
```

cParameter.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cParameter"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cParameter.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a parameter.
'               Contains functions to insert, update and delete
'               workspace_parameters records from the database.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
Option Base 0

' Local variable(s) to hold property value(s)
Private mlngWorkspaceId As Long
Private mlngParameterId As Long
Private mstrParameterName As String
```

```
Private mstrParameterValue As String
Private mstrDescription As String
Private mintParameterType As Integer
Private mdbsStepMaster As Database
Private mintOperation As Operation
Private mlngPosition As Long

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cParameter."

' The cSequence class is used to generate unique parameter identifiers
Private mParameterSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM

' Parameter types
Public Enum ParameterType
    gintParameterGeneric = 0
    gintParameterConnect
    gintParameterApplication
    gintParameterBuiltIn
End Enum


Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the field names. When the parameter names are
    ' the same as the field names, parameters in the where
    ' clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[w_id]"
                prmParam.Value = mlngWorkspaceId

            Case "[p_id]"
                prmParam.Value = mlngParameterId

            Case "[p_name]"
                prmParam.Value = mstrParameterName

            Case "[p_value]"
                prmParam.Value = mstrParameterValue

            Case "[desc]"
                prmParam.Value = mstrDescription

            Case "[p_type]"
                prmParam.Value = mintParameterType

            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource, _
                        prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrModuleName &
"AssignParameters", _
                        LoadResString(errInvalidParameter)
        End Select
    Next prmParam

'    qyExec.Parameters("w_id").Value = mlngWorkspaceId
'    qyExec.Parameters("p_id").Value = mlngParameterId
'    qyExec.Parameters("p_name").Value = mstrParameterName
'    qyExec.Parameters("p_value").Value = mstrParameterValue
'

    Exit Sub

AssignParametersErr:

    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrSource, LoadResString(errAssignParametersFailed)

End Sub

Public Property Let Position(ByVal RHS As Long)

    mlngPosition = RHS

End Property

Public Property Get Position() As Long

    Position = mlngPosition

End Property

Public Function Clone() As cParameter

    ' Creates a copy of a given parameter

    Dim cCloneParam As cParameter

    On Error GoTo CloneErr
    mstrSource = mstrModuleName & "Clone"

    Set cCloneParam = New cParameter

    ' Copy all the parameter properties to the newly
    ' created parameter
```

```
    Set cCloneParam.NodeDB = mdbsStepMaster
    cCloneParam.WorkspaceId = mlngWorkspaceId
    cCloneParam.ParameterId = mlngParameterId
    cCloneParam.ParameterName = mstrParameterName
    cCloneParam.ParameterValue = mstrParameterValue
    cCloneParam.Description = mstrDescription
    cCloneParam.ParameterType = mintParameterType
    cCloneParam.IndOperation = mintOperation
    cCloneParam.Position = mlngPosition

    ' And set the return value to the newly created parameter
    Set Clone = cCloneParam
    Set cCloneParam = Nothing

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, _
            mstrSource, LoadResString(errCloneFailed)

End Function
Public Property Set NodeDB(vdata As Database)

    Set mdbsStepMaster = vdata

End Property
Public Property Get NodeDB() As Database

    Set NodeDB = mdbsStepMaster

End Property

Private Sub CheckDupParameterName()
    ' Check if the parameter name already exists in the workspace

    Dim rstParameter As Recordset
    Dim strSql As String
    Dim qy As DAO.QueryDef

    On Error GoTo CheckDupParameterNameErr
    mstrSource = mstrModuleName & "CheckDupParameterName"

    ' Create a recordset object to retrieve the count of all parameters
    ' for the workspace with the same name
    strSql = "Select count(*) as parameter_count " & _
        " from workspace_parameters " & _
        " where workspace_id = [w_id]" & _
        " and parameter_name = [p_name]" & _
        " and parameter_id <> [p_id]"

    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strSql)
    Call AssignParameters(qy)

    Set rstParameter = qy.OpenRecordset(dbOpenForwardOnly)

    If rstParameter![parameter_count] > 0 Then
        rstParameter.Close
        qy.Close
        ShowError errDuplicateParameterName
        On Error GoTo 0
        Err.Raise vbObjectError + errDuplicateParameterName, _
            mstrSource, LoadResString(errDuplicateParameterName)
    End If

    rstParameter.Close
    qy.Close

    Exit Sub

CheckDupParameterNameErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "CheckDupParameterName"
    On Error GoTo 0
    Err.Raise vbObjectError + errCheckDupParameterNameFailed, _
        mstrSource, LoadResString(errCheckDupParameterNameFailed)

End Sub
Private Sub CheckDB()
    ' Check if the database object has been initialized

    If mdbsStepMaster Is Nothing Then
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidDB, _
            mstrModuleName & "CheckDB", LoadResString(errInvalidDB)
    End If

End Sub
Public Property Let ParameterValue(vdata As String)

    mstrParameterValue = vdata

End Property
Public Property Let Description(vdata As String)

    mstrDescription = vdata

End Property
Public Property Let ParameterType(vdata As ParameterType)

    mintParameterType = vdata

End Property

Public Property Let ParameterName(vdata As String)

    If vdata = gstrEmptyString Then

        ShowError errParameterNameMandatory
        On Error GoTo 0
```

```
        ' Propogate this error back to the caller
        Err.Raise vbObjectError + errParameterNameMandatory, _
            mstrSource, LoadResString(errParameterNameMandatory)
    Else
        mstrParameterName = vdata
    End If

End Property

Public Property Let ParameterId(vdata As Long)
    mlngParameterId = vdata
End Property

Public Property Let IndOperation(ByVal vdata As Operation)

    ' The valid operations are define in the cOperations
    ' class. Check if the operation is valid
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errInvalidOperation, _
                    mstrSource, LoadResString(errInvalidOperation)
    End Select

End Property
Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    On Error GoTo ValidateErr

    ' Check if the db object is valid
    Call CheckDB

    ' Call procedure to raise an error if the parameter name
    ' already exists in the workspace -
    ' if there are duplicates, we don't know what value for the
    ' parameter to use at runtime
    Call CheckDupParameterName

    Exit Sub

ValidateErr:

    mstrSource = mstrModuleName & "Validate"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errValidateFailed, _
        mstrSource, LoadResString(errValidateFailed)

End Sub
Public Property Let WorkspaceId(vdata As Long)

    mlngWorkspaceId = vdata

End Property

Public Sub Add()

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddErr

    ' Validate the record before trying to insert the record
    Call Validate

    ' Create a temporary querydef object
    strInsert = "insert into workspace_parameters " & _
            "( workspace_id, parameter_id, " & _
            " parameter_name, parameter_value, " & _
            " description, parameter_type ) " & _
            " values ( [w_id], [p_id], [p_name], [p_value], [desc],
[p_type] ) "
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to assign the parameter values
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close

'    strInsert = "insert into workspace_parameters " & _
'            "( workspace_id, parameter_id, " & _
'            " parameter_name, parameter_value ) " & _
'            " values ( " & _
'            Str(mlngWorkspaceId) & ", " & Str(mlngParameterId) & _
'            ", " & mFieldValue.MakeStringFieldValid(mstrParameterName)
& _
'            ", " & mFieldValue.MakeStringFieldValid(mstrParameterValue)
& " ) "
'    mdbsStepMaster.Execute strInsert, dbFailOnError + dbSQLPassThrough

    Exit Sub

AddErr:

    mstrSource = mstrModuleName & "Add"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errParameterInsertFailed, _
        mstrSource, LoadResString(errParameterInsertFailed)

End Sub
Public Sub Delete()
```

```
    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr

    ' Check if the db object is valid
    Call CheckDB

    strDelete = "delete from workspace_parameters " & _
            " where parameter_id = [p_id]"
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteParameterFailed, _
            mstrSource, _
            LoadResString(errDeleteParameterFailed)

End Sub

Public Sub Modify()

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo ModifyErr

    ' Validate the updated values before trying to modify the db
    Call Validate

    ' Create a temporary querydef object with the modify string
    strUpdate = "update workspace_parameters " & _
            " set workspace_id = [w_id], " & _
            "parameter_name = [p_name], " & _
            "parameter_value = [p_value], " & _
            "description = [desc], " & _
            "parameter_type = [p_type] " & _
            " where parameter_id = [p_id]"
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the parameter values to the
    ' querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

'    mdbsStepMaster.Execute strUpdate, dbFailOnError
'
    Exit Sub

ModifyErr:

    mstrSource = mstrModuleName & "Modify"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errParameterUpdateFailed, _
        mstrSource, LoadResString(errParameterUpdateFailed)

End Sub
Public Property Get ParameterName() As String

    ParameterName = mstrParameterName

End Property

Public Property Get ParameterId() As Long

    ParameterId = mlngParameterId

End Property
Public Property Get NextIdentifier() As Long

    Dim lngNextId As Long

    On Error GoTo NextIdentifierErr

    ' First check if the database object is valid
    Call CheckDB

    ' Retrieve the next identifier using the sequence class
    Set mParameterSeq = New cSequence
    Set mParameterSeq.IdDatabase = mdbsStepMaster
    mParameterSeq.IdentifierColumn = FLD_ID_PARAMETER
    lngNextId = mParameterSeq.Identifier
    Set mParameterSeq = Nothing

    NextIdentifier = lngNextId
    Exit Property

NextIdentifierErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "NextIdentifier"
    On Error GoTo 0
    Err.Raise vbObjectError + errIdGetFailed, _
        mstrSource, LoadResString(errIdGetFailed)

End Property
Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property
```

```
Public Property Get WorkspaceId() As Long

    WorkspaceId = mlngWorkspaceId

End Property

Public Property Get ParameterValue() As String

    ParameterValue = mstrParameterValue

End Property
Public Property Get Description() As String

    Description = mstrDescription

End Property
Public Property Get ParameterType() As ParameterType

    ParameterType = mintParameterType

End Property


Private Sub Class_Initialize()

    Set mFieldValue = New cStringSM

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

End Sub

Private Sub Class_Terminate()

    Set mdbsStepMaster = Nothing
    Set mFieldValue = Nothing

End Sub
```

cRunColIt.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunColIt"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cRunColIt.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module implements a stack of Iterator nodes.
'               Ensures that only cRunItNode objects are stored in the
stack.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cRunColIt."
Private mstrSource As String

Private mcIterators As cStack
Public Sub Clear()

    mcIterators.Clear

End Sub

Private Sub Class_Initialize()

    Set mcIterators = New cStack

End Sub

Private Sub Class_Terminate()

    Set mcIterators = Nothing

End Sub


Public Function Value(strItName As String) As String

    Dim lngIndex As Long

    For lngIndex = 0 To mcIterators.Count - 1
        If mcIterators(lngIndex).IteratorName = strItName Then
            Value = mcIterators(lngIndex).Value
            Exit For
        End If
    Next lngIndex

End Function

Public Property Get Item(ByVal Position As Long) As cRunItNode
Attribute Item.VB_UserMemId = 0

    Set Item = mcIterators(Position)

End Property
```

```
Public Function Count() As Long

    Count = mcIterators.Count

End Function

Public Function Pop() As cRunItNode

    Set Pop = mcIterators.Pop

End Function

Public Sub Push(objToPush As cRunItNode)

    Call mcIterators.Push(objToPush)

End Sub
```

cRunInst.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cRunInst"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cRunColIt.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module controls the run processing. It runs a
branch
'               at a time and raises events when each step completes
execution.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cRunInst."
Private mstrSource As String

' Local variable(s) to hold property value(s)
Private mstrRootKey As String
Public WspId As Long
Private mcParameters As cArrParameters
Private mcRunSteps As cArrSteps
Private mcRunConstraints As cArrConstraints
Public RunConnections As cConnections
Public RunConnDtls As cConnDtls
Private mcvntWspPreCons As Variant
Private mcvntWspPostCons As Variant
Private mcNavSteps As cStepTree

Private mcInstances As cInstances
Private mcFreeSteps As cVectorLng
Private mcFailures As cFailedSteps
Private mblnAsk As Boolean   ' Set to True when the a step with
continuation criteria=Ask fails
Private mblnAbort As Boolean ' Set to True when the run is aborted
Private msAbortDtls As String
Private mbarrFree() As Byte
Private WithEvents mcTermSteps As cTermSteps
Attribute mcTermSteps.VB_VarHelpID = -1
Public RunId As Long
Public CreateInputFiles As Boolean

Private Enum WspLogEvents
    mintRunStart
    mintRunComplete
    mintStepStart
    mintStepComplete
End Enum

Private mcWspLog As cFileSM

Private mstrCurBranchRoot As String
Private mcDummyRootInstance As cInstance

' Key for the dummy root instance - Should be a key that is invalid for
an actual step record
Private Const mstrDummyRootKey As String = "D"

' Public events to notify the calling function of the
' start and end time for each step
Public Event RunStart(dtmStartTime As Currency, strWspLog As String)
Public Event RunComplete(dtmEndTime As Currency)
Public Event StepStart(cStepRecord As cStep, dtmStartTime As Currency, _
        lngInstanceId As Long, lParentInstanceId As Long, sPath As
String, _
        sIts As String, sItValue As String)
Public Event StepComplete(cStepRecord As cStep, dtmEndTime As Currency,
lngInstanceId As Long, lElapsed As Long)
Public Event ProcessStart(cStepRecord As cStep, strCommand As String, _
        dtmStartTime As Currency, lngInstanceId As Long,
lParentInstanceId As Long, _
        sItValue As String)
Public Event ProcessComplete(cStepRecord As cStep, dtmEndTime As
Currency, lngInstanceId As Long, lElapsed As Long)

' The class that will execute each step - we trap the events
```

```
' that are raised by it when a step starts/completes
' execution
Private WithEvents cExecStep1 As cRunStep
Attribute cExecStep1.VB_VarHelpID = -1
Private WithEvents cExecStep2 As cRunStep
Attribute cExecStep2.VB_VarHelpID = -1
Private WithEvents cExecStep3 As cRunStep
Attribute cExecStep3.VB_VarHelpID = -1
Private WithEvents cExecStep4 As cRunStep
Attribute cExecStep4.VB_VarHelpID = -1
Private WithEvents cExecStep5 As cRunStep
Attribute cExecStep5.VB_VarHelpID = -1
Private WithEvents cExecStep6 As cRunStep
Attribute cExecStep6.VB_VarHelpID = -1
Private WithEvents cExecStep7 As cRunStep
Attribute cExecStep7.VB_VarHelpID = -1
Private WithEvents cExecStep8 As cRunStep
Attribute cExecStep8.VB_VarHelpID = -1
Private WithEvents cExecStep9 As cRunStep
Attribute cExecStep9.VB_VarHelpID = -1

Private WithEvents cExecStep10 As cRunStep
Attribute cExecStep10.VB_VarHelpID = -1
Private WithEvents cExecStep11 As cRunStep
Attribute cExecStep11.VB_VarHelpID = -1
Private WithEvents cExecStep12 As cRunStep
Attribute cExecStep12.VB_VarHelpID = -1
Private WithEvents cExecStep13 As cRunStep
Attribute cExecStep13.VB_VarHelpID = -1
Private WithEvents cExecStep14 As cRunStep
Attribute cExecStep14.VB_VarHelpID = -1
Private WithEvents cExecStep15 As cRunStep
Attribute cExecStep15.VB_VarHelpID = -1
Private WithEvents cExecStep16 As cRunStep
Attribute cExecStep16.VB_VarHelpID = -1
Private WithEvents cExecStep17 As cRunStep
Attribute cExecStep17.VB_VarHelpID = -1
Private WithEvents cExecStep18 As cRunStep
Attribute cExecStep18.VB_VarHelpID = -1
Private WithEvents cExecStep19 As cRunStep
Attribute cExecStep19.VB_VarHelpID = -1

Private WithEvents cExecStep20 As cRunStep
Attribute cExecStep20.VB_VarHelpID = -1
Private WithEvents cExecStep21 As cRunStep
Attribute cExecStep21.VB_VarHelpID = -1
Private WithEvents cExecStep22 As cRunStep
Attribute cExecStep22.VB_VarHelpID = -1
Private WithEvents cExecStep23 As cRunStep
Attribute cExecStep23.VB_VarHelpID = -1
Private WithEvents cExecStep24 As cRunStep
Attribute cExecStep24.VB_VarHelpID = -1
Private WithEvents cExecStep25 As cRunStep
Attribute cExecStep25.VB_VarHelpID = -1
Private WithEvents cExecStep26 As cRunStep
Attribute cExecStep26.VB_VarHelpID = -1
Private WithEvents cExecStep27 As cRunStep
Attribute cExecStep27.VB_VarHelpID = -1
Private WithEvents cExecStep28 As cRunStep
Attribute cExecStep28.VB_VarHelpID = -1
Private WithEvents cExecStep29 As cRunStep
Attribute cExecStep29.VB_VarHelpID = -1

Private WithEvents cExecStep30 As cRunStep
Attribute cExecStep30.VB_VarHelpID = -1
Private WithEvents cExecStep31 As cRunStep
Attribute cExecStep31.VB_VarHelpID = -1
Private WithEvents cExecStep32 As cRunStep
Attribute cExecStep32.VB_VarHelpID = -1
Private WithEvents cExecStep33 As cRunStep
Attribute cExecStep33.VB_VarHelpID = -1
Private WithEvents cExecStep34 As cRunStep
Attribute cExecStep34.VB_VarHelpID = -1
Private WithEvents cExecStep35 As cRunStep
Attribute cExecStep35.VB_VarHelpID = -1
Private WithEvents cExecStep36 As cRunStep
Attribute cExecStep36.VB_VarHelpID = -1
Private WithEvents cExecStep37 As cRunStep
Attribute cExecStep37.VB_VarHelpID = -1
Private WithEvents cExecStep38 As cRunStep
Attribute cExecStep38.VB_VarHelpID = -1
Private WithEvents cExecStep39 As cRunStep
Attribute cExecStep39.VB_VarHelpID = -1

Private WithEvents cExecStep40 As cRunStep
Attribute cExecStep40.VB_VarHelpID = -1
Private WithEvents cExecStep41 As cRunStep
Attribute cExecStep41.VB_VarHelpID = -1
Private WithEvents cExecStep42 As cRunStep
Attribute cExecStep42.VB_VarHelpID = -1
Private WithEvents cExecStep43 As cRunStep
Attribute cExecStep43.VB_VarHelpID = -1
Private WithEvents cExecStep44 As cRunStep
Attribute cExecStep44.VB_VarHelpID = -1
Private WithEvents cExecStep45 As cRunStep
Attribute cExecStep45.VB_VarHelpID = -1
Private WithEvents cExecStep46 As cRunStep
Attribute cExecStep46.VB_VarHelpID = -1
Private WithEvents cExecStep47 As cRunStep
Attribute cExecStep47.VB_VarHelpID = -1
Private WithEvents cExecStep48 As cRunStep
Attribute cExecStep48.VB_VarHelpID = -1
Private WithEvents cExecStep49 As cRunStep
Attribute cExecStep49.VB_VarHelpID = -1

Private WithEvents cExecStep50 As cRunStep
Attribute cExecStep50.VB_VarHelpID = -1
Private WithEvents cExecStep51 As cRunStep
Attribute cExecStep51.VB_VarHelpID = -1
Private WithEvents cExecStep52 As cRunStep
Attribute cExecStep52.VB_VarHelpID = -1
Private WithEvents cExecStep53 As cRunStep
```

```
Attribute cExecStep53.VB_VarHelpID = -1
Private WithEvents cExecStep54 As cRunStep
Attribute cExecStep54.VB_VarHelpID = -1
Private WithEvents cExecStep55 As cRunStep
Attribute cExecStep55.VB_VarHelpID = -1
Private WithEvents cExecStep56 As cRunStep
Attribute cExecStep56.VB_VarHelpID = -1
Private WithEvents cExecStep57 As cRunStep
Attribute cExecStep57.VB_VarHelpID = -1
Private WithEvents cExecStep58 As cRunStep
Attribute cExecStep58.VB_VarHelpID = -1
Private WithEvents cExecStep59 As cRunStep
Attribute cExecStep59.VB_VarHelpID = -1

Private WithEvents cExecStep60 As cRunStep
Attribute cExecStep60.VB_VarHelpID = -1
Private WithEvents cExecStep61 As cRunStep
Attribute cExecStep61.VB_VarHelpID = -1
Private WithEvents cExecStep62 As cRunStep
Attribute cExecStep62.VB_VarHelpID = -1
Private WithEvents cExecStep63 As cRunStep
Attribute cExecStep63.VB_VarHelpID = -1
Private WithEvents cExecStep64 As cRunStep
Attribute cExecStep64.VB_VarHelpID = -1
Private WithEvents cExecStep65 As cRunStep
Attribute cExecStep65.VB_VarHelpID = -1
Private WithEvents cExecStep66 As cRunStep
Attribute cExecStep66.VB_VarHelpID = -1
Private WithEvents cExecStep67 As cRunStep
Attribute cExecStep67.VB_VarHelpID = -1
Private WithEvents cExecStep68 As cRunStep
Attribute cExecStep68.VB_VarHelpID = -1
Private WithEvents cExecStep69 As cRunStep
Attribute cExecStep69.VB_VarHelpID = -1

Private WithEvents cExecStep70 As cRunStep
Attribute cExecStep70.VB_VarHelpID = -1
Private WithEvents cExecStep71 As cRunStep
Attribute cExecStep71.VB_VarHelpID = -1
Private WithEvents cExecStep72 As cRunStep
Attribute cExecStep72.VB_VarHelpID = -1
Private WithEvents cExecStep73 As cRunStep
Attribute cExecStep73.VB_VarHelpID = -1
Private WithEvents cExecStep74 As cRunStep
Attribute cExecStep74.VB_VarHelpID = -1
Private WithEvents cExecStep75 As cRunStep
Attribute cExecStep75.VB_VarHelpID = -1
Private WithEvents cExecStep76 As cRunStep
Attribute cExecStep76.VB_VarHelpID = -1
Private WithEvents cExecStep77 As cRunStep
Attribute cExecStep77.VB_VarHelpID = -1
Private WithEvents cExecStep78 As cRunStep
Attribute cExecStep78.VB_VarHelpID = -1
Private WithEvents cExecStep79 As cRunStep
Attribute cExecStep79.VB_VarHelpID = -1
Private WithEvents cExecStep80 As cRunStep
Attribute cExecStep80.VB_VarHelpID = -1
Private WithEvents cExecStep81 As cRunStep
Attribute cExecStep81.VB_VarHelpID = -1
Private WithEvents cExecStep82 As cRunStep
Attribute cExecStep82.VB_VarHelpID = -1
Private WithEvents cExecStep83 As cRunStep
Attribute cExecStep83.VB_VarHelpID = -1
Private WithEvents cExecStep84 As cRunStep
Attribute cExecStep84.VB_VarHelpID = -1
Private WithEvents cExecStep85 As cRunStep
Attribute cExecStep85.VB_VarHelpID = -1
Private WithEvents cExecStep86 As cRunStep
Attribute cExecStep86.VB_VarHelpID = -1
Private WithEvents cExecStep87 As cRunStep
Attribute cExecStep87.VB_VarHelpID = -1
Private WithEvents cExecStep88 As cRunStep
Attribute cExecStep88.VB_VarHelpID = -1
Private WithEvents cExecStep89 As cRunStep
Attribute cExecStep89.VB_VarHelpID = -1

Private WithEvents cExecStep90 As cRunStep
Attribute cExecStep90.VB_VarHelpID = -1
Private WithEvents cExecStep91 As cRunStep
Attribute cExecStep91.VB_VarHelpID = -1
Private WithEvents cExecStep92 As cRunStep
Attribute cExecStep92.VB_VarHelpID = -1
Private WithEvents cExecStep93 As cRunStep
Attribute cExecStep93.VB_VarHelpID = -1
Private WithEvents cExecStep94 As cRunStep
Attribute cExecStep94.VB_VarHelpID = -1
Private WithEvents cExecStep95 As cRunStep
Attribute cExecStep95.VB_VarHelpID = -1
Private WithEvents cExecStep96 As cRunStep
Attribute cExecStep96.VB_VarHelpID = -1
Private WithEvents cExecStep97 As cRunStep
Attribute cExecStep97.VB_VarHelpID = -1
Private WithEvents cExecStep98 As cRunStep
Attribute cExecStep98.VB_VarHelpID = -1
Private WithEvents cExecStep99 As cRunStep
Attribute cExecStep99.VB_VarHelpID = -1

Private Const msIt As String = " Iterator: "
Private Const msItValue As String = " Value: "
Public Sub Abort()

    On Error GoTo AbortErr

    ' Make sure that we don't execute any more steps
    Call StopRun

    If cExecStep1 Is Nothing And cExecStep2 Is Nothing And cExecStep3 Is
Nothing And cExecStep4 Is Nothing And cExecStep5 Is Nothing And
cExecStep6 Is Nothing And cExecStep7 Is Nothing And cExecStep8 Is
Nothing And cExecStep9 Is Nothing And _
```

```
       cExecStep10 Is Nothing And cExecStep11 Is Nothing And cExecStep12
Is Nothing And cExecStep13 Is Nothing And cExecStep14 Is Nothing And
cExecStep15 Is Nothing And cExecStep16 Is Nothing And cExecStep17 Is
Nothing And cExecStep18 Is Nothing And cExecStep19 Is Nothing And _
       cExecStep20 Is Nothing And cExecStep21 Is Nothing And cExecStep22
Is Nothing And cExecStep23 Is Nothing And cExecStep24 Is Nothing And
cExecStep25 Is Nothing And cExecStep26 Is Nothing And cExecStep27 Is
Nothing And cExecStep28 Is Nothing And cExecStep29 Is Nothing And _
       cExecStep30 Is Nothing And cExecStep31 Is Nothing And cExecStep32
Is Nothing And cExecStep33 Is Nothing And cExecStep34 Is Nothing And
cExecStep35 Is Nothing And cExecStep36 Is Nothing And cExecStep37 Is
Nothing And cExecStep38 Is Nothing And cExecStep39 Is Nothing And _
       cExecStep40 Is Nothing And cExecStep41 Is Nothing And cExecStep42
Is Nothing And cExecStep43 Is Nothing And cExecStep44 Is Nothing And
cExecStep45 Is Nothing And cExecStep46 Is Nothing And cExecStep47 Is
Nothing And cExecStep48 Is Nothing And cExecStep49 Is Nothing And _
       cExecStep50 Is Nothing And cExecStep51 Is Nothing And cExecStep52
Is Nothing And cExecStep53 Is Nothing And cExecStep54 Is Nothing And
cExecStep55 Is Nothing And cExecStep56 Is Nothing And cExecStep57 Is
Nothing And cExecStep58 Is Nothing And cExecStep59 Is Nothing And _
       cExecStep60 Is Nothing And cExecStep61 Is Nothing And cExecStep62
Is Nothing And cExecStep63 Is Nothing And cExecStep64 Is Nothing And
cExecStep65 Is Nothing And cExecStep66 Is Nothing And cExecStep67 Is
Nothing And cExecStep68 Is Nothing And cExecStep69 Is Nothing And _
       cExecStep70 Is Nothing And cExecStep71 Is Nothing And cExecStep72
Is Nothing And cExecStep73 Is Nothing And cExecStep74 Is Nothing And
cExecStep75 Is Nothing And cExecStep76 Is Nothing And cExecStep77 Is
Nothing And cExecStep78 Is Nothing And cExecStep79 Is Nothing And _
       cExecStep80 Is Nothing And cExecStep81 Is Nothing And cExecStep82
Is Nothing And cExecStep83 Is Nothing And cExecStep84 Is Nothing And
cExecStep85 Is Nothing And cExecStep86 Is Nothing And cExecStep87 Is
Nothing And cExecStep88 Is Nothing And cExecStep89 Is Nothing And _
       cExecStep90 Is Nothing And cExecStep91 Is Nothing And cExecStep92
Is Nothing And cExecStep93 Is Nothing And cExecStep94 Is Nothing And
cExecStep95 Is Nothing And cExecStep96 Is Nothing And cExecStep97 Is
Nothing And cExecStep98 Is Nothing And cExecStep99 Is Nothing Then
        ' Then...
        WriteToWspLog (mintRunComplete)
        RaiseEvent RunComplete(Determine64BitTime())
    Else
        ' Abort each of the steps that is currently executing.
        If Not cExecStep1 Is Nothing Then
            cExecStep1.Abort
        End If

        If Not cExecStep2 Is Nothing Then
            cExecStep2.Abort
        End If

        If Not cExecStep3 Is Nothing Then
            cExecStep3.Abort
        End If

        If Not cExecStep4 Is Nothing Then
            cExecStep4.Abort
        End If
```

```
If Not cExecStep5 Is Nothing Then
    cExecStep5.Abort
End If

If Not cExecStep6 Is Nothing Then
    cExecStep6.Abort
End If

If Not cExecStep7 Is Nothing Then
    cExecStep7.Abort
End If

If Not cExecStep8 Is Nothing Then
    cExecStep8.Abort
End If

If Not cExecStep9 Is Nothing Then
    cExecStep9.Abort
End If

If Not cExecStep10 Is Nothing Then
    cExecStep10.Abort
End If

If Not cExecStep11 Is Nothing Then
    cExecStep11.Abort
End If

If Not cExecStep12 Is Nothing Then
    cExecStep12.Abort
End If

If Not cExecStep13 Is Nothing Then
    cExecStep13.Abort
End If

If Not cExecStep14 Is Nothing Then
    cExecStep14.Abort
End If

If Not cExecStep15 Is Nothing Then
    cExecStep15.Abort
End If

If Not cExecStep16 Is Nothing Then
    cExecStep16.Abort
End If

If Not cExecStep17 Is Nothing Then
    cExecStep17.Abort
End If

If Not cExecStep18 Is Nothing Then
    cExecStep18.Abort
End If
```

```
If Not cExecStep19 Is Nothing Then
    cExecStep19.Abort
End If

If Not cExecStep20 Is Nothing Then
    cExecStep20.Abort
End If

If Not cExecStep21 Is Nothing Then
    cExecStep21.Abort
End If

If Not cExecStep22 Is Nothing Then
    cExecStep22.Abort
End If

If Not cExecStep23 Is Nothing Then
    cExecStep23.Abort
End If

If Not cExecStep24 Is Nothing Then
    cExecStep24.Abort
End If

If Not cExecStep25 Is Nothing Then
    cExecStep25.Abort
End If

If Not cExecStep26 Is Nothing Then
    cExecStep26.Abort
End If

If Not cExecStep27 Is Nothing Then
    cExecStep27.Abort
End If

If Not cExecStep28 Is Nothing Then
    cExecStep28.Abort
End If

If Not cExecStep29 Is Nothing Then
    cExecStep29.Abort
End If

' ============== 30 - 39 ==============
If Not cExecStep30 Is Nothing Then
    cExecStep30.Abort
End If

If Not cExecStep31 Is Nothing Then
    cExecStep31.Abort
End If

If Not cExecStep32 Is Nothing Then
    cExecStep32.Abort
End If

If Not cExecStep33 Is Nothing Then
    cExecStep33.Abort
End If

If Not cExecStep34 Is Nothing Then
    cExecStep34.Abort
End If

If Not cExecStep35 Is Nothing Then
    cExecStep35.Abort
End If

If Not cExecStep36 Is Nothing Then
    cExecStep36.Abort
End If

If Not cExecStep37 Is Nothing Then
    cExecStep37.Abort
End If

If Not cExecStep38 Is Nothing Then
    cExecStep38.Abort
End If

If Not cExecStep39 Is Nothing Then
    cExecStep39.Abort
End If

' ============== 40 - 49 ==============
If Not cExecStep40 Is Nothing Then
    cExecStep40.Abort
End If

If Not cExecStep41 Is Nothing Then
    cExecStep41.Abort
End If

If Not cExecStep42 Is Nothing Then
    cExecStep42.Abort
End If

If Not cExecStep43 Is Nothing Then
    cExecStep43.Abort
End If

If Not cExecStep44 Is Nothing Then
    cExecStep44.Abort
End If

If Not cExecStep45 Is Nothing Then
    cExecStep45.Abort
End If

If Not cExecStep46 Is Nothing Then
```

```
    cExecStep46.Abort
End If

If Not cExecStep47 Is Nothing Then
    cExecStep47.Abort
End If

If Not cExecStep48 Is Nothing Then
    cExecStep48.Abort
End If

If Not cExecStep49 Is Nothing Then
    cExecStep49.Abort
End If

' ============== 50 - 59 ==============
If Not cExecStep50 Is Nothing Then
    cExecStep50.Abort
End If

If Not cExecStep51 Is Nothing Then
    cExecStep51.Abort
End If

If Not cExecStep52 Is Nothing Then
    cExecStep52.Abort
End If

If Not cExecStep53 Is Nothing Then
    cExecStep53.Abort
End If

If Not cExecStep54 Is Nothing Then
    cExecStep54.Abort
End If

If Not cExecStep55 Is Nothing Then
    cExecStep55.Abort
End If

If Not cExecStep56 Is Nothing Then
    cExecStep56.Abort
End If

If Not cExecStep57 Is Nothing Then
    cExecStep57.Abort
End If

If Not cExecStep58 Is Nothing Then
    cExecStep58.Abort
End If

If Not cExecStep59 Is Nothing Then
    cExecStep59.Abort
End If

' ============== 60 - 69 ==============
If Not cExecStep60 Is Nothing Then
    cExecStep60.Abort
End If

If Not cExecStep61 Is Nothing Then
    cExecStep61.Abort
End If

If Not cExecStep62 Is Nothing Then
    cExecStep62.Abort
End If

If Not cExecStep63 Is Nothing Then
    cExecStep63.Abort
End If

If Not cExecStep64 Is Nothing Then
    cExecStep64.Abort
End If

If Not cExecStep65 Is Nothing Then
    cExecStep65.Abort
End If

If Not cExecStep66 Is Nothing Then
    cExecStep66.Abort
End If

If Not cExecStep67 Is Nothing Then
    cExecStep67.Abort
End If

If Not cExecStep68 Is Nothing Then
    cExecStep68.Abort
End If

If Not cExecStep69 Is Nothing Then
    cExecStep69.Abort
End If

' ============== 70 - 79 ==============
If Not cExecStep70 Is Nothing Then
    cExecStep70.Abort
End If

If Not cExecStep71 Is Nothing Then
    cExecStep71.Abort
End If

If Not cExecStep72 Is Nothing Then
    cExecStep72.Abort
End If

If Not cExecStep73 Is Nothing Then
    cExecStep73.Abort
```

```
End If

If Not cExecStep74 Is Nothing Then
    cExecStep74.Abort
End If

If Not cExecStep75 Is Nothing Then
    cExecStep75.Abort
End If

If Not cExecStep76 Is Nothing Then
    cExecStep76.Abort
End If

If Not cExecStep77 Is Nothing Then
    cExecStep77.Abort
End If

If Not cExecStep78 Is Nothing Then
    cExecStep78.Abort
End If

If Not cExecStep79 Is Nothing Then
    cExecStep79.Abort
End If

' ============== 80 - 89 ==============
If Not cExecStep80 Is Nothing Then
    cExecStep80.Abort
End If

If Not cExecStep81 Is Nothing Then
    cExecStep81.Abort
End If

If Not cExecStep82 Is Nothing Then
    cExecStep82.Abort
End If

If Not cExecStep83 Is Nothing Then
    cExecStep83.Abort
End If

If Not cExecStep84 Is Nothing Then
    cExecStep84.Abort
End If

If Not cExecStep85 Is Nothing Then
    cExecStep85.Abort
End If

If Not cExecStep86 Is Nothing Then
    cExecStep86.Abort
End If

If Not cExecStep87 Is Nothing Then
    cExecStep87.Abort
End If

If Not cExecStep88 Is Nothing Then
    cExecStep88.Abort
End If

If Not cExecStep89 Is Nothing Then
    cExecStep89.Abort
End If

' ============== 90 - 99 ==============
If Not cExecStep90 Is Nothing Then
    cExecStep90.Abort
End If

If Not cExecStep91 Is Nothing Then
    cExecStep91.Abort
End If

If Not cExecStep92 Is Nothing Then
    cExecStep92.Abort
End If

If Not cExecStep93 Is Nothing Then
    cExecStep93.Abort
End If

If Not cExecStep94 Is Nothing Then
    cExecStep94.Abort
End If

If Not cExecStep95 Is Nothing Then
    cExecStep95.Abort
End If

If Not cExecStep96 Is Nothing Then
    cExecStep96.Abort
End If

If Not cExecStep97 Is Nothing Then
    cExecStep97.Abort
End If

If Not cExecStep98 Is Nothing Then
    cExecStep98.Abort
End If

If Not cExecStep99 Is Nothing Then
    cExecStep99.Abort
End If

End If

Exit Sub
```

```
AbortErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    ShowError errAbortFailed
    ' Try to abort the remaining steps, if any
    Resume Next

End Sub
Public Sub AbortSiblings(cTermInstance As cInstance)

    On Error GoTo AbortSiblingsErr

    ' Abort each of the steps that is currently executing.
    If Not cExecStep1 Is Nothing Then
        If cExecStep1.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep1.Abort
        End If
    End If

    If Not cExecStep2 Is Nothing Then
        If cExecStep2.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep2.Abort
        End If
    End If

    If Not cExecStep3 Is Nothing Then
        If cExecStep3.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep3.Abort
        End If
    End If

    If Not cExecStep4 Is Nothing Then
        If cExecStep4.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep4.Abort
        End If
    End If

    If Not cExecStep5 Is Nothing Then
        If cExecStep5.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep5.Abort
        End If
    End If

    If Not cExecStep6 Is Nothing Then
        If cExecStep6.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep6.Abort
        End If
    End If

    If Not cExecStep7 Is Nothing Then
        If cExecStep7.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep7.Abort
        End If
    End If

    If Not cExecStep8 Is Nothing Then
        If cExecStep8.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep8.Abort
        End If
    End If

    If Not cExecStep9 Is Nothing Then
        If cExecStep9.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep9.Abort
        End If
    End If

    If Not cExecStep10 Is Nothing Then
        If cExecStep10.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep10.Abort
        End If
    End If

    If Not cExecStep11 Is Nothing Then
        If cExecStep11.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep11.Abort
        End If
    End If

    If Not cExecStep12 Is Nothing Then
        If cExecStep12.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep12.Abort
        End If
    End If

    If Not cExecStep13 Is Nothing Then
        If cExecStep13.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep13.Abort
        End If
    End If

    If Not cExecStep14 Is Nothing Then
        If cExecStep14.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep14.Abort
        End If
    End If

    If Not cExecStep15 Is Nothing Then
```

```
        If cExecStep15.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep15.Abort
        End If
    End If

    If Not cExecStep16 Is Nothing Then
        If cExecStep16.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep16.Abort
        End If
    End If

    If Not cExecStep17 Is Nothing Then
        If cExecStep17.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep17.Abort
        End If
    End If

    If Not cExecStep18 Is Nothing Then
        If cExecStep18.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep18.Abort
        End If
    End If

    If Not cExecStep19 Is Nothing Then
        If cExecStep19.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep19.Abort
        End If
    End If

    If Not cExecStep20 Is Nothing Then
        If cExecStep20.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep20.Abort
        End If
    End If

    If Not cExecStep21 Is Nothing Then
        If cExecStep21.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep21.Abort
        End If
    End If

    If Not cExecStep22 Is Nothing Then
        If cExecStep22.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep22.Abort
        End If
    End If

    If Not cExecStep23 Is Nothing Then
        If cExecStep23.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep23.Abort
        End If
    End If

    If Not cExecStep24 Is Nothing Then
        If cExecStep24.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep24.Abort
        End If
    End If

    If Not cExecStep25 Is Nothing Then
        If cExecStep25.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep25.Abort
        End If
    End If

    If Not cExecStep26 Is Nothing Then
        If cExecStep26.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep26.Abort
        End If
    End If

    If Not cExecStep27 Is Nothing Then
        If cExecStep27.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep27.Abort
        End If
    End If

    If Not cExecStep28 Is Nothing Then
        If cExecStep28.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep28.Abort
        End If
    End If

    If Not cExecStep29 Is Nothing Then
        If cExecStep29.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep29.Abort
        End If
    End If

    ' ============ 30 =================
    If Not cExecStep30 Is Nothing Then
        If cExecStep30.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep30.Abort
        End If
    End If
```

```
    If Not cExecStep31 Is Nothing Then
        If cExecStep31.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep31.Abort
        End If
    End If

    If Not cExecStep32 Is Nothing Then
        If cExecStep32.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep32.Abort
        End If
    End If

    If Not cExecStep33 Is Nothing Then
        If cExecStep33.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep33.Abort
        End If
    End If

    If Not cExecStep34 Is Nothing Then
        If cExecStep34.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep34.Abort
        End If
    End If

    If Not cExecStep35 Is Nothing Then
        If cExecStep35.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep35.Abort
        End If
    End If

    If Not cExecStep36 Is Nothing Then
        If cExecStep36.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep36.Abort
        End If
    End If

    If Not cExecStep37 Is Nothing Then
        If cExecStep37.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep37.Abort
        End If
    End If

    If Not cExecStep38 Is Nothing Then
        If cExecStep38.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep38.Abort
        End If
    End If

    If Not cExecStep39 Is Nothing Then
        If cExecStep39.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep39.Abort
        End If
    End If

    ' ============ 40 =================
    If Not cExecStep40 Is Nothing Then
        If cExecStep40.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep40.Abort
        End If
    End If

    If Not cExecStep41 Is Nothing Then
        If cExecStep41.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep41.Abort
        End If
    End If

    If Not cExecStep42 Is Nothing Then
        If cExecStep42.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep42.Abort
        End If
    End If

    If Not cExecStep43 Is Nothing Then
        If cExecStep43.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep43.Abort
        End If
    End If

    If Not cExecStep44 Is Nothing Then
        If cExecStep44.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep44.Abort
        End If
    End If

    If Not cExecStep45 Is Nothing Then
        If cExecStep45.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep45.Abort
        End If
    End If

    If Not cExecStep46 Is Nothing Then
        If cExecStep46.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep46.Abort
        End If
    End If
```

```
    If Not cExecStep47 Is Nothing Then
        If cExecStep47.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep47.Abort
        End If
    End If

    If Not cExecStep48 Is Nothing Then
        If cExecStep48.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep48.Abort
        End If
    End If

    If Not cExecStep49 Is Nothing Then
        If cExecStep49.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep49.Abort
        End If
    End If

    ' ============ 50 =================
    If Not cExecStep50 Is Nothing Then
        If cExecStep50.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep50.Abort
        End If
    End If

    If Not cExecStep51 Is Nothing Then
        If cExecStep51.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep51.Abort
        End If
    End If

    If Not cExecStep52 Is Nothing Then
        If cExecStep52.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep52.Abort
        End If
    End If

    If Not cExecStep53 Is Nothing Then
        If cExecStep53.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep53.Abort
        End If
    End If

    If Not cExecStep54 Is Nothing Then
        If cExecStep54.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep54.Abort
        End If
    End If

    If Not cExecStep55 Is Nothing Then
        If cExecStep55.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep55.Abort
        End If
    End If

    If Not cExecStep56 Is Nothing Then
        If cExecStep56.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep56.Abort
        End If
    End If

    If Not cExecStep57 Is Nothing Then
        If cExecStep57.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep57.Abort
        End If
    End If

    If Not cExecStep58 Is Nothing Then
        If cExecStep58.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep58.Abort
        End If
    End If

    If Not cExecStep59 Is Nothing Then
        If cExecStep59.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep59.Abort
        End If
    End If

    ' ============ 60 =================
    If Not cExecStep60 Is Nothing Then
        If cExecStep60.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep60.Abort
        End If
    End If

    If Not cExecStep61 Is Nothing Then
        If cExecStep61.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep61.Abort
        End If
    End If

    If Not cExecStep62 Is Nothing Then
        If cExecStep62.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep62.Abort
```

```
        End If
    End If

    If Not cExecStep63 Is Nothing Then
        If cExecStep63.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep63.Abort
        End If
    End If

    If Not cExecStep64 Is Nothing Then
        If cExecStep64.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep64.Abort
        End If
    End If

    If Not cExecStep65 Is Nothing Then
        If cExecStep65.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep65.Abort
        End If
    End If

    If Not cExecStep66 Is Nothing Then
        If cExecStep66.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep66.Abort
        End If
    End If

    If Not cExecStep67 Is Nothing Then
        If cExecStep67.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep67.Abort
        End If
    End If

    If Not cExecStep68 Is Nothing Then
        If cExecStep68.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep68.Abort
        End If
    End If

    If Not cExecStep69 Is Nothing Then
        If cExecStep69.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep69.Abort
        End If
    End If

    ' ============ 70 =================
    If Not cExecStep70 Is Nothing Then
        If cExecStep70.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep70.Abort
        End If
    End If

    If Not cExecStep71 Is Nothing Then
        If cExecStep71.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep71.Abort
        End If
    End If

    If Not cExecStep72 Is Nothing Then
        If cExecStep72.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep72.Abort
        End If
    End If

    If Not cExecStep73 Is Nothing Then
        If cExecStep73.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep73.Abort
        End If
    End If

    If Not cExecStep74 Is Nothing Then
        If cExecStep74.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep74.Abort
        End If
    End If

    If Not cExecStep75 Is Nothing Then
        If cExecStep75.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep75.Abort
        End If
    End If

    If Not cExecStep76 Is Nothing Then
        If cExecStep76.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep76.Abort
        End If
    End If

    If Not cExecStep77 Is Nothing Then
        If cExecStep77.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep77.Abort
        End If
    End If

    If Not cExecStep78 Is Nothing Then
        If cExecStep78.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
```

```
            cExecStep78.Abort
        End If
    End If

    If Not cExecStep79 Is Nothing Then
        If cExecStep79.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep79.Abort
        End If
    End If

    ' ============ 80 =================
    If Not cExecStep80 Is Nothing Then
        If cExecStep80.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep80.Abort
        End If
    End If

    If Not cExecStep81 Is Nothing Then
        If cExecStep81.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep81.Abort
        End If
    End If

    If Not cExecStep82 Is Nothing Then
        If cExecStep82.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep82.Abort
        End If
    End If

    If Not cExecStep83 Is Nothing Then
        If cExecStep83.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep83.Abort
        End If
    End If

    If Not cExecStep84 Is Nothing Then
        If cExecStep84.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep84.Abort
        End If
    End If

    If Not cExecStep85 Is Nothing Then
        If cExecStep85.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep85.Abort
        End If
    End If

    If Not cExecStep86 Is Nothing Then
        If cExecStep86.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep86.Abort
        End If
    End If

    If Not cExecStep87 Is Nothing Then
        If cExecStep87.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep87.Abort
        End If
    End If

    If Not cExecStep88 Is Nothing Then
        If cExecStep88.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep88.Abort
        End If
    End If

    If Not cExecStep89 Is Nothing Then
        If cExecStep89.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep89.Abort
        End If
    End If

    ' ============ 90 =================
    If Not cExecStep90 Is Nothing Then
        If cExecStep90.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep90.Abort
        End If
    End If

    If Not cExecStep91 Is Nothing Then
        If cExecStep91.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep91.Abort
        End If
    End If

    If Not cExecStep92 Is Nothing Then
        If cExecStep92.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep92.Abort
        End If
    End If

    If Not cExecStep93 Is Nothing Then
        If cExecStep93.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep93.Abort
        End If
    End If
```

```
    If Not cExecStep94 Is Nothing Then
        If cExecStep94.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep94.Abort
        End If
    End If

    If Not cExecStep95 Is Nothing Then
        If cExecStep95.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep95.Abort
        End If
    End If

    If Not cExecStep96 Is Nothing Then
        If cExecStep96.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep96.Abort
        End If
    End If

    If Not cExecStep97 Is Nothing Then
        If cExecStep97.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep97.Abort
        End If
    End If

    If Not cExecStep98 Is Nothing Then
        If cExecStep98.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep98.Abort
        End If
    End If

    If Not cExecStep99 Is Nothing Then
        If cExecStep99.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep99.Abort
        End If
    End If

    Exit Sub

AbortSiblingsErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    ShowError errAbortFailed
    ' Try to abort the remaining steps, if any
    Resume Next

End Sub
Private Sub ExecutionFailed(cTermStep As cRunStep)
    ' Called when execution of a step fails for any reason - ensure that
execution
    ' continues
    On Error GoTo ExecutionFailedErr

    Call AddFreeProcess(cTermStep.Index)

    Call RunBranch(mstrCurBranchRoot)

    Exit Sub

ExecutionFailedErr:
    ' Log the error code raised by Visual Basic - do not raise an error
here!
    Call LogErrors(Errors)

End Sub
Private Sub FreeExecStep(lngIndex As Long)
    ' Frees an instance of a cExecuteSM object depending on the index
    On Error GoTo FreeExecStepErr

    Select Case lngIndex + 1
        Case 1
            Set cExecStep1 = Nothing
        Case 2
            Set cExecStep2 = Nothing
        Case 3
            Set cExecStep3 = Nothing
        Case 4
            Set cExecStep4 = Nothing
        Case 5
            Set cExecStep5 = Nothing
        Case 6
            Set cExecStep6 = Nothing
        Case 7
            Set cExecStep7 = Nothing
        Case 8
            Set cExecStep8 = Nothing
        Case 9
            Set cExecStep9 = Nothing
        Case 10
            Set cExecStep10 = Nothing
        Case 11
            Set cExecStep11 = Nothing
        Case 12
            Set cExecStep12 = Nothing
        Case 13
            Set cExecStep13 = Nothing
        Case 14
            Set cExecStep14 = Nothing
        Case 15
            Set cExecStep15 = Nothing
        Case 16
            Set cExecStep16 = Nothing
        Case 17
            Set cExecStep17 = Nothing
        Case 18
            Set cExecStep18 = Nothing
```

```
Case 19
    Set cExecStep19 = Nothing
Case 20
    Set cExecStep20 = Nothing
Case 21
    Set cExecStep21 = Nothing
Case 22
    Set cExecStep22 = Nothing
Case 23
    Set cExecStep23 = Nothing
Case 24
    Set cExecStep24 = Nothing
Case 25
    Set cExecStep25 = Nothing
Case 26
    Set cExecStep26 = Nothing
Case 27
    Set cExecStep27 = Nothing
Case 28
    Set cExecStep28 = Nothing
Case 29
    Set cExecStep29 = Nothing
Case 30
    Set cExecStep30 = Nothing
Case 31
    Set cExecStep31 = Nothing
Case 32
    Set cExecStep32 = Nothing
Case 33
    Set cExecStep33 = Nothing
Case 34
    Set cExecStep34 = Nothing
Case 35
    Set cExecStep35 = Nothing
Case 36
    Set cExecStep36 = Nothing
Case 37
    Set cExecStep37 = Nothing
Case 38
    Set cExecStep38 = Nothing
Case 39
    Set cExecStep39 = Nothing
Case 40
    Set cExecStep40 = Nothing
Case 41
    Set cExecStep41 = Nothing
Case 42
    Set cExecStep42 = Nothing
Case 43
    Set cExecStep43 = Nothing
Case 44
    Set cExecStep44 = Nothing
Case 45
    Set cExecStep45 = Nothing
Case 46
    Set cExecStep46 = Nothing
Case 47
    Set cExecStep47 = Nothing
Case 48
    Set cExecStep48 = Nothing
Case 49
    Set cExecStep49 = Nothing
Case 50
    Set cExecStep50 = Nothing
Case 51
    Set cExecStep51 = Nothing
Case 52
    Set cExecStep52 = Nothing
Case 53
    Set cExecStep53 = Nothing
Case 54
    Set cExecStep54 = Nothing
Case 55
    Set cExecStep55 = Nothing
Case 56
    Set cExecStep56 = Nothing
Case 57
    Set cExecStep57 = Nothing
Case 58
    Set cExecStep58 = Nothing
Case 59
    Set cExecStep59 = Nothing
Case 60
    Set cExecStep60 = Nothing
Case 61
    Set cExecStep61 = Nothing
Case 62
    Set cExecStep62 = Nothing
Case 63
    Set cExecStep63 = Nothing
Case 64
    Set cExecStep64 = Nothing
Case 65
    Set cExecStep65 = Nothing
Case 66
    Set cExecStep66 = Nothing
Case 67
    Set cExecStep67 = Nothing
Case 68
    Set cExecStep68 = Nothing
Case 69
    Set cExecStep69 = Nothing
Case 70
    Set cExecStep70 = Nothing
Case 71
    Set cExecStep71 = Nothing
Case 72
    Set cExecStep72 = Nothing
Case 73
    Set cExecStep73 = Nothing
Case 74
    Set cExecStep74 = Nothing
```

```
    Case 75
        Set cExecStep75 = Nothing
    Case 76
        Set cExecStep76 = Nothing
    Case 77
        Set cExecStep77 = Nothing
    Case 78
        Set cExecStep78 = Nothing
    Case 79
        Set cExecStep79 = Nothing
    Case 80
        Set cExecStep80 = Nothing
    Case 81
        Set cExecStep81 = Nothing
    Case 82
        Set cExecStep82 = Nothing
    Case 83
        Set cExecStep83 = Nothing
    Case 84
        Set cExecStep84 = Nothing
    Case 85
        Set cExecStep85 = Nothing
    Case 86
        Set cExecStep86 = Nothing
    Case 87
        Set cExecStep87 = Nothing
    Case 88
        Set cExecStep88 = Nothing
    Case 89
        Set cExecStep89 = Nothing
    Case 90
        Set cExecStep90 = Nothing
    Case 91
        Set cExecStep91 = Nothing
    Case 92
        Set cExecStep92 = Nothing
    Case 93
        Set cExecStep93 = Nothing
    Case 94
        Set cExecStep94 = Nothing
    Case 95
        Set cExecStep95 = Nothing
    Case 96
        Set cExecStep96 = Nothing
    Case 97
        Set cExecStep97 = Nothing
    Case 98
        Set cExecStep98 = Nothing
    Case 99
        Set cExecStep99 = Nothing
    Case Else
        BugAssert False, "FreeExecStep: Invalid index value!"
End Select

Exit Sub

FreeExecStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)

End Sub
Private Sub ProcessAskFailures()
    ' This procedure is called when a step with a continuation criteria
= Ask has failed.
    ' Wait for all running processes to complete before displaying an
Abort/Retry/Fail
    ' message to the user. We process every Ask step that has failed and
use a simple
    ' algorithm to determine what to do next.
    ' 1. An abort response to any failure results in an immediate abort
of the run
    ' 2. A continue means the run continues - this failure is popped off
the failure list.
    ' 3. A retry means that the execution details for the instance are
cleared and the
    '    step is re-executed.
    Dim lIndex As Long
    Dim cStepRec As cStep
    Dim cNextInst As cInstance
    Dim cFailureRec As cFailedStep

    On Error GoTo ProcessAskFailuresErr

    ' Display a popup message for all steps that have failed with a
continuation
    ' criteria of Ask
    For lIndex = mcFailures.Count - 1 To 0 Step -1

        Set cFailureRec = mcFailures(lIndex)

        If cFailureRec.ContCriteria = gintOnFailureAsk Then
            Set cStepRec = mcRunSteps.QueryStep(cFailureRec.StepId)
            ' Ask the user whether to abort/retry/continue
            #If RUN_ONLY Then
                cFailureRec.AskResponse = ShowMessageBox(0, _
                        "Step '" & GetStepNodeText(cStepRec) & "' failed.
" & _
                        "Select Abort to abort run and Ignore to
continue. " & _
                        "Select Retry to re-execute the failed step.", _
                        "Step Failure", _
                        MB_ABORTRETRYIGNORE + MB_APPLMODAL +
MB_ICONEXCLAMATION)
            #Else
                cFailureRec.AskResponse = ShowMessageBox(frmRunning.hWnd,
_
                        "Step '" & GetStepNodeText(cStepRec) & "' failed.
" & _
                        "Select Abort to abort run and Ignore to
continue. " & _
                        "Select Retry to re-execute the failed step.", _
                        "Step Failure", _
```

```
                        MB_ABORTRETRYIGNORE + MB_APPLMODAL +
MB_ICONEXCLAMATION)
            #End If

            ' Process an abort response immediately
            If cFailureRec.AskResponse = IDABORT Then
                mblnAbort = True
                Set cNextInst =
mcInstances.QueryInstance(cFailureRec.InstanceId)
                Call RunPendingSiblings(cNextInst, cFailureRec.EndTime)
                Exit For
            End If
        End If

    Next lIndex

    ' Process all failed steps for which we have Ignore and Retry
responses.
    If Not mblnAbort Then
        ' Navigate in reverse order since we'll be deleting items from
the collection
        For lIndex = mcFailures.Count - 1 To 0 Step -1
            If mcFailures(lIndex).ContCriteria = gintOnFailureAsk Then
                mblnAsk = False
                Set cFailureRec = mcFailures.Delete(lIndex)

                Select Case cFailureRec.AskResponse
                    Case IDABORT
                        BugAssert True

                    Case IDRETRY
                        ' Delete all instances for the failed step and
re-try
                        ' Returns a parent instance reference
                        Set cNextInst = ProcessRetryStep(cFailureRec)
                        Call RunPendingStepInBranch(mstrCurBranchRoot,
cNextInst)

                    Case IDIGNORE
                        Set cNextInst =
mcInstances.QueryInstance(cFailureRec.InstanceId)
                        Call RunPendingSiblings(cNextInst,
cFailureRec.EndTime)

                End Select
            End If
        Next lIndex
    End If

    Exit Sub

ProcessAskFailuresErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errExecuteBranchFailed, mstrModuleName, _
            LoadResString(errExecuteBranchFailed)
End Sub
Private Function ProcessRetryStep(cFailureRec As cFailedStep) As
cInstance
    ' This procedure is called when a step with a continuation criteria
= Ask has failed
    ' and the user wants to re-execute the step.
    ' We delete all existing instances for the step and reset the
iterator, if
    ' any on the parent instance - this way we ensure that the step will
be executed
    ' in the next pass.
    Dim lIndex As Long
    Dim cParentInstance As cInstance
    Dim cSubStepRec As cSubStep
    Dim cStepRec As cStep

    On Error GoTo ProcessRetryStepErr

    ' Navigate in reverse order since we'll be deleting items from the
collection
    For lIndex = mcInstances.Count - 1 To 0 Step -1

        If mcInstances(lIndex).Step.StepId = cFailureRec.StepId Then
            Set cParentInstance =
mcInstances.QueryInstance(mcInstances(lIndex).ParentInstanceId)
            Set cSubStepRec =
cParentInstance.QuerySubStep(cFailureRec.StepId)
            Set cStepRec = mcRunSteps.QueryStep(cFailureRec.StepId)

            ' Decrement the child count on the parent instance and reset
the
            ' step iterators on the sub-step record, if any -
            ' all the iterations of the step will be re-executed.
            cParentInstance.ChildDeleted cFailureRec.StepId
            cParentInstance.AllComplete = False
            cParentInstance.AllStarted = False

            cSubStepRec.InitializeIt cStepRec, mcParameters

            ' Now delete the current instance
            Set ProcessRetryStep = mcInstances.Delete(lIndex)
        End If
    Next lIndex

    Exit Function

ProcessRetryStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errExecuteBranchFailed, mstrModuleName, _
            LoadResString(errExecuteBranchFailed)

End Function
```

```
Private Sub RunNextStep(ByVal dtmCompleteTime As Currency, ByVal
lngIndex As Long, _
        ByVal InstanceId As Long, ByVal ExecutionStatus As
InstanceStatus)
    ' Checks if there are any steps remaining to be
    ' executed in the current branch. If so, it executes
    ' the step.
    Dim cTermInstance As cInstance
    Dim cFailure As cFailedStep

    On Error GoTo RunNextStepErr

    BugMessage "RunNextStep: cExecStep" & CStr(lngIndex + 1) & " has
completed."

    Call mcTermSteps.Delete
    Call FreeExecStep(lngIndex)

    ' Call a procedure to add the freed up object to the list
    Call AddFreeProcess(lngIndex)

    Set cTermInstance = mcInstances.QueryInstance(InstanceId)
    cTermInstance.Status = ExecutionStatus

    If ExecutionStatus = gintFailed Then
        If cTermInstance.Step.ContinuationCriteria =
gintOnFailureAbortSiblings Then
            Call AbortSiblings(cTermInstance)
        End If

        If Not mcFailures.StepFailed(cTermInstance.Step.StepId) Then
            Set cFailure = New cFailedStep
            cFailure.InstanceId = cTermInstance.InstanceId
            cFailure.StepId = cTermInstance.Step.StepId
            cFailure.ParentStepId = cTermInstance.Step.ParentStepId
            cFailure.ContCriteria =
cTermInstance.Step.ContinuationCriteria
            cFailure.EndTime = dtmCompleteTime
            mcFailures.Add cFailure
            Set cFailure = Nothing
        End If
    End If

    If ExecutionStatus = gintFailed And
cTermInstance.Step.ContinuationCriteria = gintOnFailureAbort Then
        If StringEmpty(msAbortDtls) Then
            ' Initialize the abort message
            msAbortDtls = "Step '" & GetStepNodeText(cTermInstance.Step)
& "' failed. " & _
                  "Aborting execution. Please check the error file for
details."
        End If
        Call Abort
    ElseIf ExecutionStatus = gintFailed And
cTermInstance.Step.ContinuationCriteria = gintOnFailureAsk Then
        mblnAsk = True
        ' If the step failed due to a Cancel operation (Abort), abort
the run
        If mblnAbort Then
            Call RunPendingSiblings(cTermInstance, dtmCompleteTime)
        End If
    Else
        Call RunPendingSiblings(cTermInstance, dtmCompleteTime)
    End If

    If mblnAbort Then
        If Not AnyStepRunning(mcFreeSteps, mbarrFree) And Not
StringEmpty(msAbortDtls) Then
            ' Display an error only if the abort is due to a failure
            ' We had to abort since a step failed - since no other steps
are currently
            ' running, we can display a message to the user saying that
we had to abort
            #If RUN_ONLY Then
                Call ShowMessageBox(0, msAbortDtls, "Run Aborted", _
                        MB_APPLMODAL + MB_OK + MB_ICONEXCLAMATION)
            #Else
                Call ShowMessageBox(frmRunning.hWnd, msAbortDtls, "Run
Aborted", _
                        MB_APPLMODAL + MB_OK + MB_ICONEXCLAMATION)
            #End If
'            MsgBox msAbortDtls, vbOKOnly, "Run Aborted"
        End If
    ElseIf mblnAsk Then
        If Not AnyStepRunning(mcFreeSteps, mbarrFree) Then
            ' Ask the user whether to abort/retry/ignore failed steps
            Call ProcessAskFailures
        End If
    End If

    Exit Sub

RunNextStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    WriteError errExecuteBranchFailed, mstrSource
    Call ResetForm(lngIndex)

End Sub
Public Sub StopRun()

    ' Setting the Abort flag to True will ensure that we
    ' don't execute any more steps
    mblnAbort = True

End Sub

Private Sub CreateDummyInstance(strRootKey As String)

    Dim cNewInstance As cInstance
    Dim cSubStepDtls  As cStep
```

```
    Dim lngSubStepId As Long

    On Error GoTo CreateDummyInstanceErr

    ' Create a new instance of the step
    ' initialize substeps for the step
    Set cNewInstance = New cInstance

    ' There can be multiple iterations of the top level nodes
    ' running at the same time, but only one branch at any
    ' time - so enforce a degree of parallelism of 1 on this
    ' node!
    Set cNewInstance.Step = New cStep
    cNewInstance.DegreeParallelism = 1
    cNewInstance.Key = mstrDummyRootKey

    cNewInstance.InstanceId = NewInstanceId
    cNewInstance.ParentInstanceId = 0

    lngSubStepId = MakeIdentifierValid(strRootKey)

    Set cSubStepDtls = mcRunSteps.QueryStep(lngSubStepId)
    If cSubStepDtls.EnabledFlag Then
        ' Create a child node for the step corresponding to
        ' the root node of the branch being currently executed,
        ' only if it has been enabled
        Call cNewInstance.CreateSubStep(cSubStepDtls, mcParameters)
    End If

    mcInstances.Add cNewInstance
    Set cNewInstance.Iterators = DetermineIterators(cNewInstance)

    ' Set a reference to the newly created dummy instance
    Set mcDummyRootInstance = cNewInstance

    Set cNewInstance = Nothing

    Exit Sub

CreateDummyInstanceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "CreateDummyInstance"
    Err.Raise vbObjectError + errCreateInstanceFailed, _
            mstrSource, LoadResString(errCreateInstanceFailed)

End Sub
Private Function CreateInstance(cExecStep As cStep, _
        cParentInstance As cInstance) As cInstance
    ' Creates a new instance of the passed in step. Returns
    ' a reference to the newly created instance object.

    Dim cNewInstance As cInstance
    Dim nodChild As cStep
    Dim lngSubStepId As Long

    On Error GoTo CreateInstanceErr

    ' Create a new instance of the step
    ' initialize substeps for the step
    Set cNewInstance = New cInstance
    Set cNewInstance.Step = cExecStep
    cNewInstance.Key = MakeKeyValid(cExecStep.StepId,
cExecStep.StepType)
    cNewInstance.ParentInstanceId = cParentInstance.InstanceId
    cNewInstance.InstanceId = NewInstanceId
    ' Validate the degree of parallelism field before assigning it to
the instance -
    ' (the parameter value might have been set to an invalid value at
runtime)
    Call ValidateParallelism(cExecStep.DegreeParallelism, _
            cExecStep.WorkspaceId, ParamsInWsp:=mcParameters)
    cNewInstance.DegreeParallelism =
SubstituteParameters(cExecStep.DegreeParallelism, _
            cExecStep.WorkspaceId, WspParameters:=mcParameters)

    If mcNavSteps.HasChild(StepKey:=cNewInstance.Key) Then
        Set nodChild = mcNavSteps.ChildStep(StepKey:=cNewInstance.Key)
        Do
            If nodChild.EnabledFlag Then
                ' Create nodes for all it's substeps only
                ' if the substeps have been enabled
                Call cNewInstance.CreateSubStep(nodChild, mcParameters)
            End If

            Set nodChild = mcNavSteps.NextStep(StepId:=nodChild.StepId)
        Loop While (Not nodChild Is Nothing)
    End If

    mcInstances.Add cNewInstance
    Set cNewInstance.Iterators = DetermineIterators(cNewInstance)

    ' Increment the number of executing steps on the parent
    cParentInstance.ChildExecuted (cExecStep.StepId)

    Set CreateInstance = cNewInstance

    Exit Function

CreateInstanceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "CreateInstance"
    Err.Raise vbObjectError + errCreateInstanceFailed, _
            mstrSource, LoadResString(errCreateInstanceFailed)

End Function
Private Function DetermineIterators(cInstanceRec As cInstance) As
cRunColIt
    ' Returns a collection of all the iterator values for this
```

```
    ' instance - since an iterator that is defined at a
    ' particular level can be used in all it's substeps, we
    ' need to navigate the step tree all the way to the root

    Dim cRunIts As cRunColIt
    Dim cRunIt As cRunItNode
    Dim cStepIt As cIterator
    Dim cParentInst As cInstance
    Dim cSubStepRec As cSubStep
    Dim cSubStepDtls As cStep
    Dim lngSubStepId As Long
    Dim lngIndex As Long

    On Error GoTo DetermineIteratorsErr

    Set cRunIts = New cRunColIt

    If cInstanceRec.ParentInstanceId > 0 Then
        ' The last iterator for an instance of a step is stored
        ' on it's parent! So navigate up before beginning the
        ' search for iterator values.
        Set cParentInst =
mcInstances.QueryInstance(cInstanceRec.ParentInstanceId)

        ' Get the sub-step record for the current step
        ' on it's parent's instance!
        lngSubStepId = cInstanceRec.Step.StepId
        Set cSubStepRec = cParentInst.QuerySubStep(lngSubStepId)
        Set cSubStepDtls = mcRunSteps.QueryStep(lngSubStepId)

        ' And determine the next iteration value for the
        ' substep in this instance
        Set cStepIt = cSubStepRec.NewIteration(cSubStepDtls)

        If Not cStepIt Is Nothing Then
            ' Add the iterator details to the collection since
            ' an iterator has been defined for the step
            Set cRunIt = New cRunItNode
            cRunIt.IteratorName = cSubStepDtls.IteratorName
            cRunIt.Value = SubstituteParameters(cStepIt.Value,
cSubStepDtls.WorkspaceId, WspParameters:=mcParameters)
            cRunIt.StepId = cSubStepRec.StepId
            cRunIts.Push cRunIt
        End If

        ' Since the parent instance has all the iterators upto
        ' that level, read them and push them on to the stack for
        ' this instance
        For lngIndex = 0 To cParentInst.Iterators.Count - 1
            Set cRunIt = cParentInst.Iterators(lngIndex)
            cRunIts.Push cRunIt
        Next lngIndex
    End If

    Set DetermineIterators = cRunIts

    Exit Function

DetermineIteratorsErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "DetermineIterators"
    Err.Raise vbObjectError + errExecInstanceFailed, _
            mstrSource, LoadResString(errExecInstanceFailed)

End Function
Private Function DetermineConstraints(cInstanceRec As cInstance, _
        intConsType As ConstraintType) As Variant
    ' Returns a collection of all the constraints for this
    ' instance of the passed in type - all the constraints defined
    ' for the manager are executed first, followed by those defined
    ' for the step. If a step has an iterator defined for it, each
    ' constraint is executed only once.

    Dim cParentInst As cInstance
    Dim cTempInst As cInstance
    Dim vntConstraints As Variant
    Dim vntTempCons As Variant
    Dim cColConstraints() As Variant
    Dim lngConsCount As Long

    On Error GoTo DetermineConstraintsErr

    Set cTempInst = cInstanceRec
    lngConsCount = 0

    ' Go all the way to the root
    Do
        If cTempInst.ParentInstanceId > 0 Then
            Set cParentInst =
mcInstances.QueryInstance(cTempInst.ParentInstanceId)
        Else
            Set cParentInst = Nothing
        End If

        ' Check if the step has an iterator defined for it
        If cTempInst.ValidForIteration(cParentInst, intConsType) Then
            vntTempCons = mcRunConstraints.ConstraintsForStep( _
                    cTempInst.Step.StepId, cTempInst.Step.VersionNo, _
                    intConsType, blnSort:=True, _
                    blnGlobal:=False, blnGlobalConstraintsOnly:=False)

            If Not IsEmpty(vntTempCons) Then
                ReDim Preserve cColConstraints(lngConsCount)
                cColConstraints(lngConsCount) = vntTempCons
                lngConsCount = lngConsCount + 1
            End If
        End If

        Set cTempInst = cParentInst
```

```
    Loop While Not cTempInst Is Nothing

    If lngConsCount > 0 Then
        vntTempCons = OrderConstraints(cColConstraints, intConsType)
    End If

    DetermineConstraints = vntTempCons

    Exit Function

DetermineConstraintsErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "DetermineConstraints"
    Err.Raise vbObjectError + errExecInstanceFailed, _
            mstrSource, LoadResString(errExecInstanceFailed)

End Function
Private Function GetInstanceToExecute(cParentNode As cInstance, _
        cSubStepRec As cSubStep, _
        cSubStepDtls As cStep) As cInstance

    Dim cSubStepInst As cInstance

    On Error GoTo GetInstanceToExecuteErr

    BugAssert Not (cParentNode Is Nothing Or _
            cSubStepRec Is Nothing Or _
            cSubStepDtls Is Nothing), _
            "GetInstanceToExecute: Input invalid"

    ' Check if it has iterators
    If cSubStepDtls.IteratorCount = 0 Then
        ' Check if the step has been executed
        If cSubStepRec.TasksRunning = 0 And cSubStepRec.TasksComplete =
0 And _
                Not
mcInstances.CompletedInstanceExists(cParentNode.InstanceId,
cSubStepDtls) Then
            ' The sub-step hasn't been executed yet.
            ' Create an instance for it and exit
            Set cSubStepInst = CreateInstance(cSubStepDtls, cParentNode)
        Else
            Set cSubStepInst = Nothing
        End If
    Else
        ' Check if there are pending iterations for the sub-step
        If Not cSubStepRec.NextIteration(cSubStepDtls) Is Nothing Then
            ' Pending iterations exist - create an instance for the sub-
step and exit
            Set cSubStepInst = CreateInstance(cSubStepDtls, cParentNode)
        Else
            ' No more iterations - continue with the next substep
            Set cSubStepInst = Nothing
        End If
    End If

    Set GetInstanceToExecute = cSubStepInst
    Exit Function

GetInstanceToExecuteErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "GetInstanceToExecute"
    Err.Raise vbObjectError + errNavInstancesFailed, _
            mstrSource, LoadResString(errNavInstancesFailed)

End Function

Public Function InstancesForStep(lngStepId As Long, ByRef StepStatus As
InstanceStatus) As cInstances
    ' Returns an array of all the instances for a step
    Dim lngIndex As Long
    Dim cTempInst As cInstance
    Dim cStepInstances As cInstances
    Dim cStepRec As cStep

    On Error GoTo InstancesForStepErr

    Set cStepInstances = New cInstances

    For lngIndex = 0 To mcInstances.Count - 1
        Set cTempInst = mcInstances(lngIndex)

        If cTempInst.Step.StepId = lngStepId Then
            cStepInstances.Add cTempInst
        End If
    Next lngIndex

    If cStepInstances.Count = 0 Then
        Set cStepRec = mcRunSteps.QueryStep(lngStepId)
        If Not mcFailures.ExecuteSubStep(cStepRec.ParentStepId) Then
            StepStatus = gintAborted
        End If
        Set cStepRec = Nothing
    End If

    ' Set the return value of the function to the array of
    ' constraints that has been built above
    Set InstancesForStep = cStepInstances

    Set cStepInstances = Nothing
    Exit Function

InstancesForStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "InstancesForStep"
    Err.Raise vbObjectError + errNavInstancesFailed, mstrSource, _
```

```
            LoadResString(errNavInstancesFailed)

End Function
Private Sub RemoveFreeProcess(lngRunningProcess As Long)
    ' Removes the passed in element from the collection of
    ' free objects

    ' Confirm that the last element in the array is the one
    ' we need to delete
    If mcFreeSteps(mcFreeSteps.Count - 1) = lngRunningProcess Then
        mcFreeSteps.Delete Position:=mcFreeSteps.Count - 1
    Else
        ' Ask the class to find the element and delete it
        mcFreeSteps.Delete Item:=lngRunningProcess
    End If

End Sub
Private Sub AddFreeProcess(lngTerminatedProcess As Long)
    ' Adds the passed in element to the collection of
    ' free objects

    mcFreeSteps.Add lngTerminatedProcess

End Sub

Private Sub ResetForm(Optional ByVal lngIndex As Long)

    Dim lngTemp As Long

    On Error GoTo ResetFormErr

    ' Check if there are any running instances to wait for
    If mcFreeSteps.Count <> glngNumConcurrentProcesses Then

        For lngTemp = 0 To mcFreeSteps.Count - 1
            If mcFreeSteps(lngTemp) = lngIndex Then
                Exit For
            End If
        Next lngTemp

        If lngTemp <= mcFreeSteps.Count - 1 Then
            ' This process that just completed did not exist in the list
of
            ' free processes
            Call AddFreeProcess(lngIndex)
        End If

        If Not AnyStepRunning(mcFreeSteps, mbarrFree) Then
            WriteToWspLog (mintRunComplete)
            ' All steps are complete
            RaiseEvent RunComplete(Determine64BitTime())
        End If
    Else
        WriteToWspLog (mintRunComplete)
        RaiseEvent RunComplete(Determine64BitTime())
    End If

    Exit Sub

ResetFormErr:

End Sub
Private Function NewInstanceId() As Long
    ' Will return new instance id's - uses a static counter
    ' that it increments each time
    Static lngInstance As Long

    lngInstance = lngInstance + 1
    NewInstanceId = lngInstance

End Function

Private Function RunPendingStepInBranch(strCurBranchRoot As String, _
        Optional cExecInstance As cInstance = Nothing) As cInstance
    ' Runs a worker step in the branch being executed, if
    ' there are any pending execution
    ' This function is also called when a step has just completed
    ' execution - in which case the terminated instance is
    ' passed in as the optional parameter. When that happens,
    ' we first try to execute the siblings of the terminated
    ' step if any are pending execution.
    ' If the terminated instance has not been passed in, we
    ' start with the dummy root instance and navigate down,
    ' trying to find a pending worker step.

    Dim cExecSubStep As cStep
    Dim cParentInstance As cInstance
    Dim cNextInst As cInstance

    On Error GoTo RunPendingStepInBranchErr

    If Not cExecInstance Is Nothing Then
        ' Called when an instance has terminated
        ' When a worker step terminates, then we need to
        ' decremement the number of running steps on it's
        ' manager
        Set cParentInstance = _
mcInstances.QueryInstance(cExecInstance.ParentInstanceId)

    Else
        If StringEmpty(strCurBranchRoot) Or mcDummyRootInstance Is
Nothing Then
            ' Run complete - event raised by Run method
            Set RunPendingStepInBranch = Nothing
            Exit Function
        End If

        ' If there are no pending steps on the root instance,
        ' then there are no steps within the branch that need
        ' to be executed
```

```
        If mcDummyRootInstance.AllComplete Or
mcDummyRootInstance.AllStarted Then
            Set RunPendingStepInBranch = Nothing
            Exit Function
        End If

        Set cParentInstance = mcDummyRootInstance
    End If

    Do
        Set cNextInst = GetSubStepToExecute(cParentInstance)
        If cNextInst Is Nothing Then
            ' There are no steps within the branch that can
            ' be executed - If we are at the dummy instance,
            ' this branch has completed executing
            If cParentInstance.Key = mstrDummyRootKey Then
                Set cNextInst = Nothing
                Exit Do
            Else
                ' Go to the parent instance and try to find
                ' some other sibling is pending execution
                Set cNextInst =
mcInstances.QueryInstance(cParentInstance.ParentInstanceId)

                If cParentInstance.SubSteps.Count = 0 Then
                    cNextInst.ChildTerminated
cParentInstance.Step.StepId
                End If
            End If
        End If

        BugAssert Not cNextInst Is Nothing
        Set cParentInstance = cNextInst

    Loop While cNextInst.Step.StepType <> gintWorkerStep

    If Not cNextInst Is Nothing Then
        Call ExecuteStep(cNextInst)
    End If

    Set RunPendingStepInBranch = cNextInst

    Exit Function

RunPendingStepInBranchErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errNavInstancesFailed, _
            mstrModuleName & "RunPendingStepInBranch",
LoadResString(errNavInstancesFailed)

End Function
Private Function RunPendingSibling(cTermInstance As cInstance, _
        dtmCompleteTime As Currency) As cInstance
    ' This process is called when a step terminates. Tries to
    ' run a sibling of the terminated step, if one is pending
    ' execution.

    Dim cParentInstance As cInstance
    Dim cNextInst As cInstance

    On Error GoTo RunPendingSiblingErr

    If StringEmpty(mstrCurBranchRoot) Or mcDummyRootInstance Is Nothing
Then
        ' Run complete - event raised by Run method
        Set RunPendingSibling = Nothing
        Exit Function
    End If

    BugAssert cTermInstance.ParentInstanceId > 0, "Orphaned instance in
array!"

    ' When a worker step terminates, then we need to
    ' decremement the number of running steps on it's
    ' manager
    Set cParentInstance =
mcInstances.QueryInstance(cTermInstance.ParentInstanceId)

    ' Decrement the number of running processes on the
    ' parent by 1
    Call cParentInstance.ChildTerminated(cTermInstance.Step.StepId)

    ' The first step that terminates has to be a worker
    ' If it is complete, update the completed steps on the
    ' parent by 1.
    Call cParentInstance.ChildCompleted(cTermInstance.Step.StepId)
    cParentInstance.AllStarted = False

    Do
        Set cNextInst = GetSubStepToExecute(cParentInstance,
dtmCompleteTime)
        If cNextInst Is Nothing Then
            If cParentInstance.Key = mstrDummyRootKey Then
                Set cNextInst = Nothing
                Exit Do
            Else
                ' Go to the parent instance and try to find
                ' some other sibling is pending execution
                Set cNextInst =
mcInstances.QueryInstance(cParentInstance.ParentInstanceId)
                If cParentInstance.IsRunning Then
                    cNextInst.AllStarted = True
                Else
                    ' No more sub-steps to execute
                    Call
cNextInst.ChildCompleted(cParentInstance.Step.StepId)
                    Call
cNextInst.ChildTerminated(cParentInstance.Step.StepId)
                    cNextInst.AllStarted = False
                End If
```

```
            End If
        End If

        BugAssert Not cNextInst Is Nothing
        Set cParentInstance = cNextInst

    Loop While cNextInst.Step.StepType <> gintWorkerStep

    If Not cNextInst Is Nothing Then
        Call ExecuteStep(cNextInst)
    End If

    Set RunPendingSibling = cNextInst

    Exit Function

RunPendingSiblingErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "RunPendingSibling"
    Err.Raise vbObjectError + errNavInstancesFailed, mstrSource, _
        LoadResString(errNavInstancesFailed)

End Function
Private Sub RunPendingSiblings(cTermInstance As cInstance, _
        dtmCompleteTime As Currency)
    ' This process is called when a step terminates. Tries to
    ' run siblings of the terminated step, if they are pending
    ' execution.

    Dim cExecInst As cInstance

    On Error GoTo RunPendingSiblingsErr
    BugMessage "In RunPendingSiblings"

    ' Call a procedure to run the sibling of the terminated
    ' step, if any. This procedure will also update the
    ' number of complete/running tasks on the manager steps.
    Set cExecInst = RunPendingSibling(cTermInstance, dtmCompleteTime)

    If Not cExecInst Is Nothing Then
        Do
            ' Execute any other pending steps in the branch.
            ' The step that has just terminated might be
            ' the last one that was executing in a sub-branch.
            ' That would mean that we can execute another
            ' sub-branch that might involve more than 1 step.
            ' Pass the just executed step as a parameter.
            Set cExecInst = RunPendingStepInBranch(mstrCurBranchRoot,
cExecInst)
        Loop While Not cExecInst Is Nothing
    Else
        If Not mcDummyRootInstance.IsRunning Then
            ' All steps have been executed in the branch - run
            ' a new branch
            Call RunNewBranch
        Else
            ' There are no more steps to execute in the current
            ' branch but we have running processes.
        End If
    End If

    Exit Sub

RunPendingSiblingsErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "RunPendingSiblings"
    Err.Raise vbObjectError + errNavInstancesFailed, _
            mstrSource, LoadResString(errNavInstancesFailed)

End Sub

Private Sub NoSubStepsToExecute(cMgrInstance As cInstance, Optional
dtmCompleteTime As Currency = gdtmEmpty)
    ' Called when we cannot find any more substeps to run for
    ' manager step - set the allcomplete or allstarted
    ' properties to true

    If cMgrInstance.IsRunning() Then
        cMgrInstance.AllStarted = True
    Else
        cMgrInstance.AllComplete = True
        If dtmCompleteTime <> gdtmEmpty Then
            ' Update the end time on the manager step
            Call TimeCompleteUpdateForStep(cMgrInstance,
dtmCompleteTime)
        End If
    End If

End Sub

Private Function GetSubStepToExecute(cParentNode As cInstance, _
        Optional dtmCompleteTime As Currency = 0) As cInstance
    ' Returns the child of the passed in node that is to be
    ' executed next. Checks if we are in the middle of an instance
    ' being executed in which case it returns the pending
    ' instance. Creates a new instance if there are pending
    ' instances for a sub-step.

    Dim lngIndex As Long
    Dim cSubStepRec As cSubStep
    Dim cSubStepDtls As cStep
    Dim cSubStepInst As cInstance

    On Error GoTo GetSubStepToExecuteErr

    ' There are a number of cases that need to be accounted
    ' for here.
    ' 1. While traversing through all enabled nodes for the
```

```
    ' first time - instance records may not exist for the
    ' substeps.
    ' 2. Instance records exist, and there are processes
    ' that need to be executed for a sub-step
    ' 3. There are no more processes that need to be currently
    ' executed (till a process completes)
    ' 4. There are no more processes that need to be executed
    ' (All substeps have completed execution)

    ' This is the only point where we check the Abort flag -
    ' since this is the heart of the navigation routine that
    ' selects processes to execute. Also, when a step terminates
    ' selection of the next process goes through here.
    If mblnAbort Then
        Set GetSubStepToExecute = Nothing
        cParentNode.Status = gintAborted
        Exit Function
    End If

    If mblnAsk Then
        Set GetSubStepToExecute = Nothing
        Exit Function
    End If

    If Not mcFailures.ExecuteSubStep(cParentNode.Step.StepId) Then
        Set GetSubStepToExecute = Nothing
        cParentNode.Status = gintAborted
        Exit Function
    End If

    ' First check if there are pending steps for the parent!
    If cParentNode.IsPending Then
        ' Loop through all the sub-steps for the parent node
        For lngIndex = 0 To cParentNode.SubSteps.Count - 1
            Set cSubStepRec = cParentNode.SubSteps(lngIndex)
            Set cSubStepDtls = mcRunSteps.QueryStep(cSubStepRec.StepId)
            If Not mcInstances.InstanceAborted(cSubStepRec) Then
                ' Check if the sub-step is a worker
                If cSubStepDtls.StepType = gintWorkerStep Then
                    ' Find/create an instance to execute
                    Set cSubStepInst = GetInstanceToExecute( _
                            cParentNode, cSubStepRec, cSubStepDtls)
                    If Not cSubStepInst Is Nothing Then
                        Exit For
                    Else
                        ' Continue w/ the next sub-step
                    End If
                Else
                    ' The sub-step is a manager step
                    ' Check if there are any pending instances for
                    ' the manager
                    Set cSubStepInst =
mcInstances.QueryPendingInstance( _
                            cParentNode.InstanceId, cSubStepRec.StepId)
                    If cSubStepInst Is Nothing Then
                        ' Find/create an instance to execute
                        Set cSubStepInst = GetInstanceToExecute( _
                                cParentNode, cSubStepRec, cSubStepDtls)
                        If Not cSubStepInst Is Nothing Then
                            Exit For
                        Else
                            ' Continue w/ the next sub-step
                        End If
                    Else
                        ' We have found a pending instance for the
                        ' sub-step (manager) - exit the loop
                        Exit For
                    End If
                End If
            End If
        Next lngIndex

        If lngIndex > cParentNode.SubSteps.Count - 1 Or
cParentNode.SubSteps.Count = 0 Then
            ' If we could not find any sub-steps to execute,
            ' mark the parent node as complete/all started
            Call NoSubStepsToExecute(cParentNode, dtmCompleteTime)
            Set cSubStepInst = Nothing
        End If
    End If

    Set GetSubStepToExecute = cSubStepInst
    Exit Function

GetSubStepToExecuteErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "GetSubStepToExecute"
    Err.Raise vbObjectError + errNavInstancesFailed, mstrSource, _
        LoadResString(errNavInstancesFailed)

End Function

Private Sub TimeCompleteUpdateForStep(cMgrInstance As cInstance, ByVal
EndTime As Currency)

    ' Called when there are no more sub-steps to execute for
    ' the manager step. It updates the end time and status on
    ' the manager.
    Dim lElapsed As Long

    On Error GoTo TimeCompleteUpdateForStepErr

    If cMgrInstance.Key <> mstrDummyRootKey Then
        cMgrInstance.EndTime = EndTime
        cMgrInstance.Status = gintComplete
        lElapsed = (EndTime - cMgrInstance.StartTime) * 10000
        cMgrInstance.ElapsedTime = lElapsed
        RaiseEvent StepComplete(cMgrInstance.Step, EndTime,
cMgrInstance.InstanceId, lElapsed)
    End If
```

```
    Exit Sub

TimeCompleteUpdateForStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    WriteError errUpdateDisplayFailed, mstrModuleName &
"TimeCompleteUpdateForStep"

End Sub

Private Function GetFreeObject() As Long

    ' Check the array of free objects and retrieve the first one
    If mcFreeSteps.Count > 0 Then
        GetFreeObject = mcFreeSteps(mcFreeSteps.Count - 1)
    Else
        mstrSource = mstrModuleName & "GetFreeObject"
        ShowError errMaxProcessesExceeded
        On Error GoTo 0
        Err.Raise vbObjectError + errMaxProcessesExceeded, _
                mstrSource, _
                LoadResString(errMaxProcessesExceeded)
    End If

End Function
Private Function StepTerminated(cCompleteStep As cStep, ByVal
dtmCompleteTime As Currency, _
        ByVal lngIndex As Long, ByVal InstanceId As Long, ByVal
ExecutionStatus As InstanceStatus) As cStep
    ' This procedure is called whenever a step terminates.
    Dim cTermRec As cTermStep
    Dim cInstRec As cInstance
    Dim cStartInst As cInstance
    Dim lElapsed As Long
    Dim sLogLabel As String
    Dim LogLabels As New cVectorStr
    Dim iItIndex As Long

    On Error GoTo StepTerminatedErr

    Set cInstRec = mcInstances.QueryInstance(InstanceId)
    If dtmCompleteTime <> 0 And cInstRec.StartTime <> 0 Then
        ' Convert to milliseconds since that is the default precision
        lElapsed = (dtmCompleteTime - cInstRec.StartTime) * 10000
    Else
        lElapsed = 0
    End If

    Set cStartInst = cInstRec
    iItIndex = 0
    Do While cInstRec.Key <> mstrDummyRootKey
        sLogLabel = gstrSQ & cInstRec.Step.StepLabel & gstrSQ

        If iItIndex < cInstRec.Iterators.Count Then
            If cStartInst.Iterators(iItIndex).StepId =
cInstRec.Step.StepId Then
                sLogLabel = sLogLabel & msIt & gstrSQ &
cStartInst.Iterators(iItIndex).IteratorName & gstrSQ & _
                        msItValue & gstrSQ &
cStartInst.Iterators(iItIndex).Value & gstrSQ
                iItIndex = iItIndex + 1
            End If
        End If

        If cInstRec.Key = cStartInst.Key Then
            ' Append the execution status
            sLogLabel = sLogLabel & " Status: " & gstrSQ &
gsExecutionStatus(ExecutionStatus) & gstrSQ
            If ExecutionStatus = gintFailed Then
                ' Append the continuation criteria for the step since it
failed
                sLogLabel = sLogLabel & " Continuation Criteria: " &
gstrSQ & gsContCriteria(cInstRec.Step.ContinuationCriteria) & gstrSQ
            End If
        End If
        LogLabels.Add sLogLabel

        Set cInstRec =
mcInstances.QueryInstance(cInstRec.ParentInstanceId)
    Loop

    Call WriteToWspLog(mintStepComplete, LogLabels, dtmCompleteTime)
    Set LogLabels = Nothing

    ' Adds the terminated step details to a queue.
    Set cTermRec = New cTermStep
    cTermRec.ExecutionStatus = ExecutionStatus
    cTermRec.Index = lngIndex
    cTermRec.InstanceId = InstanceId
    cTermRec.TimeComplete = dtmCompleteTime
    Call mcTermSteps.Add(cTermRec)
    Set cTermRec = Nothing

    RaiseEvent StepComplete(cCompleteStep, dtmCompleteTime, InstanceId,
lElapsed)

    Exit Function

StepTerminatedErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    WriteError errExecuteBranchFailed, mstrSource
    Call ResetForm(lngIndex)

End Function
Public Property Let RootKey(ByVal vdata As String)

    mstrRootKey = vdata

End Property
```

```
Public Property Get RootKey() As String
    RootKey = mstrRootKey
End Property

Private Function InitExecStep() As cRunStep
    ' Since arrays of objects cannot be declared as WithEvents,
    ' we use a limited number of objects and set a maximum
    ' on the number of steps that can run in parallel
    ' This is a wrapper that will create an instance of
    ' a cExecuteSM object depending on the index
    Dim lngIndex As Long

    On Error GoTo InitExecStepErr

    lngIndex = GetFreeObject

    Select Case lngIndex + 1
        Case 1
            Set cExecStep1 = New cRunStep
            Set InitExecStep = cExecStep1
        Case 2
            Set cExecStep2 = New cRunStep
            Set InitExecStep = cExecStep2
        Case 3
            Set cExecStep3 = New cRunStep
            Set InitExecStep = cExecStep3
        Case 4
            Set cExecStep4 = New cRunStep
            Set InitExecStep = cExecStep4
        Case 5
            Set cExecStep5 = New cRunStep
            Set InitExecStep = cExecStep5
        Case 6
            Set cExecStep6 = New cRunStep
            Set InitExecStep = cExecStep6
        Case 7
            Set cExecStep7 = New cRunStep
            Set InitExecStep = cExecStep7
        Case 8
            Set cExecStep8 = New cRunStep
            Set InitExecStep = cExecStep8
        Case 9
            Set cExecStep9 = New cRunStep
            Set InitExecStep = cExecStep9
        Case 10
            Set cExecStep10 = New cRunStep
            Set InitExecStep = cExecStep10
        Case 11
            Set cExecStep11 = New cRunStep
            Set InitExecStep = cExecStep11
        Case 12
            Set cExecStep12 = New cRunStep
            Set InitExecStep = cExecStep12
        Case 13
            Set cExecStep13 = New cRunStep
            Set InitExecStep = cExecStep13
        Case 14
            Set cExecStep14 = New cRunStep
            Set InitExecStep = cExecStep14
        Case 15
            Set cExecStep15 = New cRunStep
            Set InitExecStep = cExecStep15
        Case 16
            Set cExecStep16 = New cRunStep
            Set InitExecStep = cExecStep16
        Case 17
            Set cExecStep17 = New cRunStep
            Set InitExecStep = cExecStep17
        Case 18
            Set cExecStep18 = New cRunStep
            Set InitExecStep = cExecStep18
        Case 19
            Set cExecStep19 = New cRunStep
            Set InitExecStep = cExecStep19
        Case 20
            Set cExecStep20 = New cRunStep
            Set InitExecStep = cExecStep20
        Case 21
            Set cExecStep21 = New cRunStep
            Set InitExecStep = cExecStep21
        Case 22
            Set cExecStep22 = New cRunStep
            Set InitExecStep = cExecStep22
        Case 23
            Set cExecStep23 = New cRunStep
            Set InitExecStep = cExecStep23
        Case 24
            Set cExecStep24 = New cRunStep
            Set InitExecStep = cExecStep24
        Case 25
            Set cExecStep25 = New cRunStep
            Set InitExecStep = cExecStep25
        Case 26
            Set cExecStep26 = New cRunStep
            Set InitExecStep = cExecStep26
        Case 27
            Set cExecStep27 = New cRunStep
            Set InitExecStep = cExecStep27
        Case 28
            Set cExecStep28 = New cRunStep
            Set InitExecStep = cExecStep28
        Case 29
            Set cExecStep29 = New cRunStep
            Set InitExecStep = cExecStep29
        Case 30
            Set cExecStep30 = New cRunStep
            Set InitExecStep = cExecStep30
        Case 31
            Set cExecStep31 = New cRunStep
            Set InitExecStep = cExecStep31
        Case 32
```

```
    Set cExecStep32 = New cRunStep
    Set InitExecStep = cExecStep32
Case 33
    Set cExecStep33 = New cRunStep
    Set InitExecStep = cExecStep33
Case 34
    Set cExecStep34 = New cRunStep
    Set InitExecStep = cExecStep34
Case 35
    Set cExecStep35 = New cRunStep
    Set InitExecStep = cExecStep35
Case 36
    Set cExecStep36 = New cRunStep
    Set InitExecStep = cExecStep36
Case 37
    Set cExecStep37 = New cRunStep
    Set InitExecStep = cExecStep37
Case 38
    Set cExecStep38 = New cRunStep
    Set InitExecStep = cExecStep38
Case 39
    Set cExecStep39 = New cRunStep
    Set InitExecStep = cExecStep39
Case 40
    Set cExecStep40 = New cRunStep
    Set InitExecStep = cExecStep40
Case 41
    Set cExecStep41 = New cRunStep
    Set InitExecStep = cExecStep41
Case 42
    Set cExecStep42 = New cRunStep
    Set InitExecStep = cExecStep42
Case 43
    Set cExecStep43 = New cRunStep
    Set InitExecStep = cExecStep43
Case 44
    Set cExecStep44 = New cRunStep
    Set InitExecStep = cExecStep44
Case 45
    Set cExecStep45 = New cRunStep
    Set InitExecStep = cExecStep45
Case 46
    Set cExecStep46 = New cRunStep
    Set InitExecStep = cExecStep46
Case 47
    Set cExecStep47 = New cRunStep
    Set InitExecStep = cExecStep47
Case 48
    Set cExecStep48 = New cRunStep
    Set InitExecStep = cExecStep48
Case 49
    Set cExecStep49 = New cRunStep
    Set InitExecStep = cExecStep49
Case 50
    Set cExecStep50 = New cRunStep
    Set InitExecStep = cExecStep50
Case 51
    Set cExecStep51 = New cRunStep
    Set InitExecStep = cExecStep51
Case 52
    Set cExecStep52 = New cRunStep
    Set InitExecStep = cExecStep52
Case 53
    Set cExecStep53 = New cRunStep
    Set InitExecStep = cExecStep53
Case 54
    Set cExecStep54 = New cRunStep
    Set InitExecStep = cExecStep54
Case 55
    Set cExecStep55 = New cRunStep
    Set InitExecStep = cExecStep55
Case 56
    Set cExecStep56 = New cRunStep
    Set InitExecStep = cExecStep56
Case 57
    Set cExecStep57 = New cRunStep
    Set InitExecStep = cExecStep57
Case 58
    Set cExecStep58 = New cRunStep
    Set InitExecStep = cExecStep58
Case 59
    Set cExecStep59 = New cRunStep
    Set InitExecStep = cExecStep59
Case 60
    Set cExecStep60 = New cRunStep
    Set InitExecStep = cExecStep60
Case 61
    Set cExecStep61 = New cRunStep
    Set InitExecStep = cExecStep61
Case 62
    Set cExecStep62 = New cRunStep
    Set InitExecStep = cExecStep62
Case 63
    Set cExecStep63 = New cRunStep
    Set InitExecStep = cExecStep63
Case 64
    Set cExecStep64 = New cRunStep
    Set InitExecStep = cExecStep64
Case 65
    Set cExecStep65 = New cRunStep
    Set InitExecStep = cExecStep65
Case 66
    Set cExecStep66 = New cRunStep
    Set InitExecStep = cExecStep66
Case 67
    Set cExecStep67 = New cRunStep
    Set InitExecStep = cExecStep67
Case 68
    Set cExecStep68 = New cRunStep
    Set InitExecStep = cExecStep68
Case 69
    Set cExecStep69 = New cRunStep
```

```
    Set InitExecStep = cExecStep69
Case 70
    Set cExecStep70 = New cRunStep
    Set InitExecStep = cExecStep70
Case 71
    Set cExecStep71 = New cRunStep
    Set InitExecStep = cExecStep71
Case 72
    Set cExecStep72 = New cRunStep
    Set InitExecStep = cExecStep72
Case 73
    Set cExecStep73 = New cRunStep
    Set InitExecStep = cExecStep73
Case 74
    Set cExecStep74 = New cRunStep
    Set InitExecStep = cExecStep74
Case 75
    Set cExecStep75 = New cRunStep
    Set InitExecStep = cExecStep75
Case 76
    Set cExecStep76 = New cRunStep
    Set InitExecStep = cExecStep76
Case 77
    Set cExecStep77 = New cRunStep
    Set InitExecStep = cExecStep77
Case 78
    Set cExecStep78 = New cRunStep
    Set InitExecStep = cExecStep78
Case 79
    Set cExecStep79 = New cRunStep
    Set InitExecStep = cExecStep79
Case 80
    Set cExecStep80 = New cRunStep
    Set InitExecStep = cExecStep80
Case 81
    Set cExecStep81 = New cRunStep
    Set InitExecStep = cExecStep81
Case 82
    Set cExecStep82 = New cRunStep
    Set InitExecStep = cExecStep82
Case 83
    Set cExecStep83 = New cRunStep
    Set InitExecStep = cExecStep83
Case 84
    Set cExecStep84 = New cRunStep
    Set InitExecStep = cExecStep84
Case 85
    Set cExecStep85 = New cRunStep
    Set InitExecStep = cExecStep85
Case 86
    Set cExecStep86 = New cRunStep
    Set InitExecStep = cExecStep86
Case 87
    Set cExecStep87 = New cRunStep
    Set InitExecStep = cExecStep87
Case 88
    Set cExecStep88 = New cRunStep
    Set InitExecStep = cExecStep88
Case 89
    Set cExecStep89 = New cRunStep
    Set InitExecStep = cExecStep89
Case 90
    Set cExecStep90 = New cRunStep
    Set InitExecStep = cExecStep90
Case 91
    Set cExecStep91 = New cRunStep
    Set InitExecStep = cExecStep91
Case 92
    Set cExecStep92 = New cRunStep
    Set InitExecStep = cExecStep92
Case 93
    Set cExecStep93 = New cRunStep
    Set InitExecStep = cExecStep93
Case 94
    Set cExecStep94 = New cRunStep
    Set InitExecStep = cExecStep94
Case 95
    Set cExecStep95 = New cRunStep
    Set InitExecStep = cExecStep95
Case 96
    Set cExecStep96 = New cRunStep
    Set InitExecStep = cExecStep96
Case 97
    Set cExecStep97 = New cRunStep
    Set InitExecStep = cExecStep97
Case 98
    Set cExecStep98 = New cRunStep
    Set InitExecStep = cExecStep98
Case 99
    Set cExecStep99 = New cRunStep
    Set InitExecStep = cExecStep99
Case Else
    Set InitExecStep = Nothing
End Select

BugMessage "Sending cExecStep" & (lngIndex + 1) & "!"

If Not InitExecStep Is Nothing Then
    InitExecStep.Index = lngIndex

    ' Remove this element from the collection of free objects
    Call RemoveFreeProcess(lngIndex)
End If

Exit Function

InitExecStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Set InitExecStep = Nothing

End Function
```

```
Public Sub Run()
    ' Calls procedures to build a list of all the steps that
    ' need to be executed and to execute them
    ' Determines whether the run has started/terminated and
    ' raises the Run Start and Complete events.
    Dim cTempStep As cStep

    On Error GoTo RunErr

    If StringEmpty(mstrRootKey) Then
        Call ShowError(errExecuteBranchFailed)
        On Error GoTo 0
        Err.Raise vbObjectError + errExecuteBranchFailed, mstrModuleName
& "Run", _
                LoadResString(errExecuteBranchFailed)
    Else
        ' Execute the first branch
        WriteToWspLog (mintRunStart)
        RaiseEvent RunStart(Determine64BitTime(), mcWspLog.FileName)

        If mcNavSteps.HasChild(StepKey:=mstrRootKey) Then
            Set cTempStep = mcNavSteps.ChildStep(StepKey:=mstrRootKey)
            mstrCurBranchRoot = MakeKeyValid(cTempStep.StepId,
cTempStep.StepType)

            Call CreateDummyInstance(mstrCurBranchRoot)

            ' Run all pending steps in the branch
            If Not RunBranch(mstrCurBranchRoot) Then
                ' Execute a new branch if there aren't any
                ' steps to run
                Call RunNewBranch
            End If
        Else
            WriteToWspLog (mintRunComplete)
            ' No children to execute - the run is complete
            RaiseEvent RunComplete(Determine64BitTime())
        End If
    End If

    Exit Sub

RunErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Call ShowError(errExecuteBranchFailed, OptArgs:=mstrCurBranchRoot)
    Call ResetForm

End Sub
Private Sub RunNewBranch()
    ' We will build a tree of all instances that occur and
    ' the count of the sub-steps that are running will be
    ' stored at each node in the tree (maintained internally
    ' as an array). Since there can be multiple iterations
    ' of the top level nodes running at the same time, we
    ' create a dummy node at the root that keeps a record of
    ' the instances of the top level node.

    ' Determines whether the run has started/terminated and
    ' raises the Run Start and Complete events.
    Dim cNextStep As cStep
    Dim bRunComplete As Boolean

    On Error GoTo RunNewBranchErr

    bRunComplete = False

    Do
        If StringEmpty(mstrCurBranchRoot) Then
            Exit Do
'            On Error GoTo 0
'            Err.Raise vbObjectError + errExecuteBranchFailed,
mstrSource, _
'                LoadResString(errExecuteBranchFailed)
        Else
            Set cNextStep =
mcNavSteps.NextStep(StepKey:=mstrCurBranchRoot)
            If cNextStep Is Nothing Then
                mstrCurBranchRoot = gstrEmptyString
                bRunComplete = True
                Exit Do
            Else
                ' Starting execution of a new branch - initialize the
                ' module-level variable
                mstrCurBranchRoot = MakeKeyValid(cNextStep.StepId,
cNextStep.StepType)
                Call CreateDummyInstance(mstrCurBranchRoot)
            End If
        End If
        Debug.Print "Running new branch: " & mstrCurBranchRoot

    ' Loop until we find a branch that has steps to execute
    Loop While Not RunBranch(mstrCurBranchRoot)

    If bRunComplete Then
        WriteToWspLog (mintRunComplete)
        ' Run is complete
        RaiseEvent RunComplete(Determine64BitTime())
    End If

    Exit Sub

RunNewBranchErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Call ShowError(errExecuteBranchFailed, OptArgs:=mstrCurBranchRoot)
    On Error GoTo 0
    mstrSource = mstrModuleName & "RunNewBranch"
    Err.Raise vbObjectError + errExecuteBranchFailed, mstrSource, _
        LoadResString(errExecuteBranchFailed)

End Sub
```

```
Private Function RunBranch(strRootNode As String) As Boolean
    ' This procedure is called to run all the necessary steps
    ' in a branch. It can also be called when a step terminates,
    ' in which case the terminated step is passed in as the
    ' optional parameter. When a step terminates, we need to
    ' either wait for some other steps to terminate before
    ' we execute more steps or run as many steps as necessary
    ' Returns True if there are steps currently executing
    ' in the branch, else returns False
    Dim cRunning As cInstance

    On Error GoTo RunBranchErr

    If Not StringEmpty(strRootNode) Then
        ' Call a procedure to execute all the enabled steps
        ' in the branch - will return the step node that is
        ' being executed - nothing means 'No more steps to
        ' execute in the branch'.
        Do
            Set cRunning = RunPendingStepInBranch(strRootNode, cRunning)

        Loop While Not cRunning Is Nothing

        RunBranch = mcDummyRootInstance.IsRunning
    End If

    Exit Function

RunBranchErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "RunBranch"
    Err.Raise vbObjectError + errExecuteBranchFailed, _
            mstrSource, LoadResString(errExecuteBranchFailed)

End Function
Private Sub TimeUpdateForProcess(StepRecord As cStep, _
        ByVal InstanceId As Long, _
        Optional ByVal StartTime As Currency = 0, _
        Optional ByVal EndTime As Currency = 0, _
        Optional ByVal ElapsedTime As Long = 0, _
        Optional Command As String)
    ' We do not maintain start and end timestamps for the constraint
    ' of a step. Hence we check if the process that just started/
    ' terminated is the worker step that is being executed. If so,
    ' we update the start/end time and status on the instance record.

    Dim cInstanceRec As cInstance
    Dim sItVal As String

    On Error GoTo TimeUpdateForProcessErr

    Set cInstanceRec = mcInstances.QueryInstance(InstanceId)

    If StartTime = 0 Then
        RaiseEvent ProcessComplete(StepRecord, EndTime, InstanceId,
ElapsedTime)
    Else
        sItVal = GetInstanceItValue(cInstanceRec)
        RaiseEvent ProcessStart(StepRecord, Command, StartTime,
InstanceId, _
                cInstanceRec.ParentInstanceId, sItVal)
    End If

    Call cInstanceRec.UpdateStartTime(StepRecord.StepId, StartTime,
EndTime, ElapsedTime)

    Exit Sub

TimeUpdateForProcessErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    WriteError errUpdateDisplayFailed, mstrModuleName &
"TimeUpdateForProcess"

End Sub
Private Sub TimeStartUpdateForStep(StepRecord As cStep, _
        ByVal InstanceId As Long, _
        ByVal StartTime As Currency)

    ' Called when a step starts execution. Checks if this is the
    ' first enabled child of the manager step. If so, updates
    ' the start time and status on the manager.
    ' Also raises the Step Start event for the completed step.

    Dim cStartInst As cInstance
    Dim cInstanceRec As cInstance
    Dim LogLabels As New cVectorStr
    Dim iItIndex As Long
    Dim sLogLabel As String
    Dim sPath As String
    Dim sIt As String
    Dim sItVal As String

    On Error GoTo TimeStartUpdateForStepErr

    Set cStartInst = mcInstances.QueryInstance(InstanceId)

    ' Determine the step path and iterator values for the step and raise
a step start event
    Set cInstanceRec = cStartInst
    Do While cInstanceRec.Key <> mstrDummyRootKey
        If Not StringEmpty(sPath) Then
            sPath = sPath & gstrFileSeparator
        End If
        sPath = sPath & gstrSQ & cInstanceRec.Step.StepLabel & gstrSQ
        Set cInstanceRec =
mcInstances.QueryInstance(cInstanceRec.ParentInstanceId)
    Loop

    For iItIndex = cStartInst.Iterators.Count - 1 To 0 Step -1
```

```
        If Not StringEmpty(sIt) Then
            sIt = sIt & gstrFileSeparator
        End If
        sIt = sIt & gstrSQ & cStartInst.Iterators(iItIndex).Value &
gstrSQ
    Next iItIndex

    sItVal = GetInstanceItValue(cStartInst)
    RaiseEvent StepStart(StepRecord, StartTime, InstanceId,
cStartInst.ParentInstanceId, _
            sPath, sIt, sItVal)

    iItIndex = 0
    Set cInstanceRec = cStartInst
    ' Raise a StepStart event for the manager step, if this is it's
first sub-step being executed
    Do While cInstanceRec.Key <> mstrDummyRootKey

        sLogLabel = gstrSQ & cInstanceRec.Step.StepLabel & gstrSQ
        If iItIndex < cStartInst.Iterators.Count Then
            If cStartInst.Iterators(iItIndex).StepId =
cInstanceRec.Step.StepId Then
                sLogLabel = sLogLabel & msIt & gstrSQ &
cStartInst.Iterators(iItIndex).IteratorName & gstrSQ & _
                        msItValue & gstrSQ &
cStartInst.Iterators(iItIndex).Value & gstrSQ
                iItIndex = iItIndex + 1
            End If
        End If
        LogLabels.Add sLogLabel

        If cInstanceRec.Key <> cStartInst.Key And cInstanceRec.StartTime
= 0 Then
            cInstanceRec.StartTime = StartTime
            cInstanceRec.Status = gintRunning
            sItVal = GetInstanceItValue(cInstanceRec)
            ' The step path and iterator values are not needed for
manager steps, since
            ' they are primarily used by the run status form
            RaiseEvent StepStart(cInstanceRec.Step, StartTime,
cInstanceRec.InstanceId, _
                    cInstanceRec.ParentInstanceId, gstrEmptyString,
gstrEmptyString, _
                    sItVal)
        End If

        Set cInstanceRec =
mcInstances.QueryInstance(cInstanceRec.ParentInstanceId)
    Loop

    Call WriteToWspLog(mintStepStart, LogLabels, StartTime)
    Set LogLabels = Nothing

    Exit Sub

TimeStartUpdateForStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    WriteError errUpdateDisplayFailed, mstrModuleName &
"TimeStartUpdateForStep"

End Sub
Private Sub WriteToWspLog(iLogEvent As WspLogEvents, Optional StepDtls
As cVectorStr, _
        Optional dtStamp As Currency = gdtmEmpty)

    ' Writes to the workspace log that is generated for the run. The
last three
    ' parameters are valid only for Step Start and Step Complete events.
    Static bError As Boolean
    Dim sLabel As String
    Dim lIndex As Long
    Dim bHdr As Boolean
    Dim cTempConn As cConnection

    On Error GoTo WriteToWspLogErr

    Select Case iLogEvent
        Case mintRunStart
            Set mcWspLog = New cFileSM
            mcWspLog.FileName = GetDefaultDir(WspId, mcParameters) &
gstrFileSeparator & _
                    Trim(Str(RunId)) & gstrFileSeparator & "SMLog-" &
Format(Now, FMT_WSP_LOG_FILE) & gstrLogFileSuffix
            mcWspLog.WriteLine (JulianDateToString(Determine64BitTime())
& " Start Run: " & vbTab & gstrSQ &
GetWorkspaceDetails(WorkspaceId:=WspId)) & gstrSQ

            ' Write all current parameter values to the log
            bHdr = False
            For lIndex = 0 To mcParameters.ParameterCount - 1
                If mcParameters(lIndex).ParameterType <>
gintParameterApplication Then
                    If Not bHdr Then
                        mcWspLog.WriteField
JulianDateToString(Determine64BitTime()) & " Parameters: "
                        bHdr = True
                    Else
                        mcWspLog.WriteField vbTab & vbTab & vbTab
                    End If
                    mcWspLog.WriteLine vbTab & gstrSQ &
mcParameters(lIndex).ParameterName & gstrSQ & vbTab & vbTab & gstrSQ &
mcParameters(lIndex).ParameterValue & gstrSQ
                End If
            Next lIndex

            ' Write all connection properties to the log
            For lIndex = 0 To RunConnections.Count - 1
                Set cTempConn = RunConnections(lIndex)
                If lIndex = 0 Then
                    mcWspLog.WriteField
JulianDateToString(Determine64BitTime()) & " Connections: "
```

```
            Else
                mcWspLog.WriteField vbTab & vbTab & vbTab
            End If
            mcWspLog.WriteLine vbTab & gstrSQ &
cTempConn.ConnectionName & gstrSQ & _
                    vbTab & vbTab & gstrSQ &
cTempConn.ConnectionValue & gstrSQ & _
                    vbTab & "No Count: " & gstrSQ &
cTempConn.NoCountDisplay & gstrSQ & gstrBlank & _
                    "No Execute: " & gstrSQ & cTempConn.NoExecute &
gstrSQ & gstrBlank & _
                    "Parse Query Only: " & gstrSQ &
cTempConn.ParseQueryOnly & gstrSQ & gstrBlank & _
                    "Quoted Identifiers: " & gstrSQ &
cTempConn.QuotedIdentifiers & gstrSQ & gstrBlank & _
                    "ANSI Nulls: " & gstrSQ & cTempConn.AnsiNulls &
gstrSQ & gstrBlank & _
                    "Show Query Plan: " & gstrSQ &
cTempConn.ShowQueryPlan & gstrSQ & gstrBlank & _
                    "Show Stats Time: " & gstrSQ &
cTempConn.ShowStatsTime & gstrSQ & gstrBlank & _
                    "Show Stats IO: " & gstrSQ &
cTempConn.ShowStatsIO & gstrSQ & gstrBlank & _
                    "Row Count" & gstrSQ & cTempConn.RowCount &
gstrSQ & gstrBlank & _
                    "Query Timeout" & gstrSQ &
cTempConn.QueryTimeOut & gstrSQ
        Next lIndex

    Case mintRunComplete
        BugAssert Not mcWspLog Is Nothing
        mcWspLog.WriteLine (JulianDateToString(Determine64BitTime())
& " Comp. Run: " & vbTab & gstrSQ &
GetWorkspaceDetails(WorkspaceId:=WspId)) & gstrSQ
        Set mcWspLog = Nothing

    Case mintStepStart
        For lIndex = StepDtls.Count - 1 To 0 Step -1
            sLabel = StepDtls(lIndex)
            If lIndex = StepDtls.Count - 1 Then
                mcWspLog.WriteLine JulianDateToString(dtStamp) & "
Start Step: " & vbTab & sLabel
            Else
                mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab &
sLabel
            End If
        Next lIndex

    Case mintStepComplete
        For lIndex = StepDtls.Count - 1 To 0 Step -1
            sLabel = StepDtls(lIndex)
            If lIndex = StepDtls.Count - 1 Then
                mcWspLog.WriteLine JulianDateToString(dtStamp) & "
Comp. Step: " & vbTab & sLabel
            Else
                mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab &
sLabel
            End If
        Next lIndex

    End Select

    Exit Sub

WriteToWspLogErr:
    If Not bError Then
        bError = True
    End If

End Sub
'Private Sub WriteToWspLog(iLogEvent As WspLogEvents, Optional StepDtls
As cVectorStr, _
'        Optional dtStamp As Date = gdtmEmpty)
'
'   This function uses the LogWriter dll - memory corruption problems
since the vb exe
'   and the vc Execute Dll both use the same dll to write.
'    ' Writes to the workspace log that is generated for the run. The
last three
'    ' parameters are valid only for StepStart and StepComplete events.
'    Static bError As Boolean
'    Static sFile As String
'    Dim sLabel As String
'    Dim lIndex As Long
'    Dim bHdr As Boolean
'
'    On Error GoTo WriteToWspLogErr
'
'    Select Case iLogEvent
'        Case mintRunStart
'            Set mcWspLog = New LOGWRITERLib.SMLog
'            sFile = App.Path & "\" & "SMLog-" & Format(Now,
FMT_WSP_LOG_FILE) & gstrLogFileSuffix
'            mcWspLog.FileName = sFile
'            mcWspLog.Init
'            mcWspLog.WriteLine (Format(Now, FMT_WSP_LOG_DATE) & " Start
Run: " & vbTab & gstrSQ & GetWorkspaceDetails(WorkspaceId:=WspId)) &
gstrSQ
'
'            ' Write all current parameter values to the log
'            bHdr = False
'            For lIndex = 0 To mcParameters.ParameterCount - 1
'                If mcParameters(lIndex).ParameterType <>
gintParameterApplication Then
'                    If Not bHdr Then
'                        'mcWspLog.WriteLine Format(Now,
FMT_WSP_LOG_DATE) & " Parameters: " & vbTab & gstrSQ &
mcParameters(lIndex).ParameterName & gstrSQ & vbTab & vbTab & gstrSQ &
mcParameters(lIndex).ParameterValue & gstrSQ
'                        bHdr = True
'                    Else
```

```
'                       'mcWspLog.WriteLine vbTab & vbTab & vbTab &
vbTab & gstrSQ & mcParameters(lIndex).ParameterName & gstrSQ & vbTab &
vbTab & gstrSQ & mcParameters(lIndex).ParameterValue & gstrSQ
'                   End If
'               End If
'           Next lIndex
'
'       Case mintRunComplete
'           BugAssert Not mcWspLog Is Nothing
'           mcWspLog.WriteLine (Format(Now, FMT_WSP_LOG_DATE) & " Comp.
Run: " & vbTab & gstrSQ & GetWorkspaceDetails(WorkspaceId:=WspId)) &
gstrSQ
'           Set mcWspLog = Nothing
'
'       Case mintStepStart
'           For lIndex = StepDtls.Count - 1 To 0 Step -1
'               sLabel = StepDtls(lIndex)
'               If lIndex = StepDtls.Count - 1 Then
'                   mcWspLog.WriteLine Format(dtStamp,
FMT_WSP_LOG_DATE) & " Start Step: " & vbTab & sLabel
'               Else
'                   mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab &
sLabel
'               End If
'           Next lIndex
'
'       Case mintStepComplete
'           For lIndex = StepDtls.Count - 1 To 0 Step -1
'               sLabel = StepDtls(lIndex)
'               If lIndex = StepDtls.Count - 1 Then
'                   mcWspLog.WriteLine Format(dtStamp,
FMT_WSP_LOG_DATE) & " Comp. Step: " & vbTab & sLabel
'               Else
'                   mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab &
sLabel
'               End If
'           Next lIndex
'
'   End Select
'
'   Exit Sub
'
'WriteToWspLogErr:
'   If Not bError Then
'       bError = True
'   End If
'
'End Sub
'
Public Property Get WspPreExecution() As Variant
    WspPreExecution = mcvntWspPreCons
End Property
Public Property Let WspPreExecution(ByVal vdata As Variant)
    mcvntWspPreCons = vdata
End Property
Public Property Get WspPostExecution() As Variant
    WspPostExecution = mcvntWspPostCons
End Property
Public Property Let WspPostExecution(ByVal vdata As Variant)
    mcvntWspPostCons = vdata
End Property

Private Sub ExecuteStep(cCurStep As cInstance)
    ' Initializes a cRunStep object with all the properties
    ' corresponding to the step to be executed and calls it's
    ' execute method to execute the step

    Dim cExecStep As cRunStep

    On Error GoTo ExecuteStepErr
    mstrSource = mstrModuleName & "ExecuteStep"

    ' Confirm that the step is a worker
    If cCurStep.Step.StepType <> gintWorkerStep Then
        On Error GoTo 0
        Err.Raise vbObjectError + errExecInstanceFailed, mstrSource, _
                LoadResString(errExecInstanceFailed)
    End If

    Set cExecStep = InitExecStep()
    ' Exceeded the number of processes that we can run simultaneously
    If cExecStep Is Nothing Then
        ' Raise an error
        On Error GoTo 0
        Err.Raise vbObjectError + errProgramError, mstrSource, _
            LoadResString(errProgramError)
    End If
    ' Initialize the instance id - not needed for step execution
    ' but necessary to identify later which instance completed
    cExecStep.InstanceId = cCurStep.InstanceId

    Set cExecStep.ExecuteStep = cCurStep.Step
    Set cExecStep.Iterators = cCurStep.Iterators
    Set cExecStep.Globals = mcRunSteps
    Set cExecStep.WspParameters = mcParameters
    Set cExecStep.WspConnections = RunConnections
    Set cExecStep.WspConnDtls = RunConnDtls

    ' Initialize all the pre and post-execution constraints that
    ' have been defined globally for the workspace
    cExecStep.WspPreCons = mcvntWspPreCons
    cExecStep.WspPostCons = mcvntWspPostCons

    ' Initialize all the pre and post-execution constraints for
    ' the step being executed
    cExecStep.PreCons = DetermineConstraints(cCurStep, gintPreStep)
    cExecStep.PostCons = DetermineConstraints(cCurStep, gintPostStep)

    cExecStep.RunId = RunId
    cExecStep.CreateInputFiles = CreateInputFiles
```

```
    ' Call the execute method to execute the step
    cExecStep.Execute

    Set cExecStep = Nothing

    Exit Sub

ExecuteStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Call ExecutionFailed(cExecStep)

End Sub

Public Property Set Steps(cRunSteps As cArrSteps)

    Set mcRunSteps = cRunSteps
    Set mcNavSteps.StepRecords = cRunSteps

End Property
Public Property Set Parameters(cParameters As cArrParameters)
    ' A reference to the parameter array - we use it to
    ' substitute parameter values in the step text

    Set mcParameters = cParameters

End Property
Public Property Get Steps() As cArrSteps

    Set Steps = mcRunSteps

End Property
Public Property Get Constraints() As cArrConstraints

    Set Constraints = mcRunConstraints

End Property
Public Property Set Constraints(vdata As cArrConstraints)

    Set mcRunConstraints = vdata

End Property


Private Sub cExecStep1_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep1_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep1_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep1.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep1_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep9_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep9_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep9_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep9.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep9_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)
```

```
    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep10_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep10_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep10_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep10.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep10_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep11_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep11_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep11_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep11.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep11_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep12_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep12_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep12_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep12.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep12_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep13_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)
```

```
End Sub


Private Sub cExecStep13_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep13_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep13.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep13_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep14_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep14_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep14_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep14.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep14_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep15_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep15_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep15_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep15.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep15_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep16_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep16_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)
```

```
End Sub


Private Sub cExecStep16_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep16.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep16_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep17_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep17_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep17_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep17.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep17_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep18_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep18_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep18_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep18.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep18_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep19_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep19_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep19_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)
```

```
    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep19.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep19_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep20_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep20_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep20_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep20.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep20_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep21_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep21_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep21_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep21.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep21_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep22_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep22_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep22_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep22.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep22_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep23_ProcessComplete(cStepRecord As cStep, _
```

```
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep23_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep23_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep23.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep23_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep24_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep24_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep24_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep24.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep24_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep25_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep25_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep25_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep25.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep25_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep26_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep26_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)
```

```
End Sub

Private Sub cExecStep26_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep26.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep26_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep27_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep27_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep27_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep27.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep27_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep28_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep28_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep28_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep28.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep28_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep29_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep29_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep29_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep29.Index,
InstanceId, Status)

End Sub
```

```
Private Sub cExecStep29_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep30_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep30_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep30_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep30.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep30_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep31_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep31_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep31_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep31.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep31_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep32_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep32_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep32_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep32.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep32_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep33_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)
```

```
End Sub

Private Sub cExecStep33_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep33_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep33.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep33_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep34_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep34_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep34_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep34.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep34_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep35_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep35_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep35_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep35.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep35_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep36_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep36_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep36_StepComplete(cStepRecord As cStep, _
```

```
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep36.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep36_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep37_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep37_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep37_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep37.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep37_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep38_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep38_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep38_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep38.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep38_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep39_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep39_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep39_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep39.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep39_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)
```

```
End Sub

Private Sub cExecStep40_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep40_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep40_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep40.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep40_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep41_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep41_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep41_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep41.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep41_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep42_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep42_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep42_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep42.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep42_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep43_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep43_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)
```

```
    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep43_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep43.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep43_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep44_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep44_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep44_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep44.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep44_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep45_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep45_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep45_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep45.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep45_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep46_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep46_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep46_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep46.Index,
InstanceId, Status)
```

```
End Sub

Private Sub cExecStep46_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep47_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep47_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep47_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep47.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep47_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep48_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep48_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep48_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep48.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep48_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep49_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep49_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep49_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep49.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep49_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep50_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)
```

```
    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep50_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep50_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep50.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep50_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep51_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep51_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep51_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep51.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep51_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep52_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep52_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep52_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep52.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep52_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep53_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep53_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub
```

```
Private Sub cExecStep53_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep53.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep53_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep54_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep54_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep54_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep54.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep54_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep55_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep55_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep55_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep55.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep55_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep56_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep56_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep56_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep56.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep56_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)
```

```
    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep57_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep57_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep57_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep57.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep57_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep58_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep58_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep58_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep58.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep58_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep59_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep59_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep59_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep59.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep59_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep60_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub
```

```
Private Sub cExecStep60_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep60_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep60.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep60_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep61_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep61_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep61_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep61.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep61_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep62_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep62_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep62_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep62.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep62_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep63_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep63_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep63_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)
```

```
    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep63.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep63_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep64_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep64_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep64_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep64.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep64_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep65_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep65_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep65_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep65.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep65_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep66_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep66_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep66_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep66.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep66_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep67_ProcessComplete(cStepRecord As cStep, _
```

```
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep67_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep67_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep67.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep67_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep68_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep68_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep68_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep68.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep68_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep69_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep69_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep69_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep69.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep69_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep70_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep70_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)
```

```
End Sub

Private Sub cExecStep70_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep70.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep70_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep71_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep71_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep71_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep71.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep71_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep72_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep72_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep72_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep72.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep72_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep73_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep73_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep73_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep73.Index,
InstanceId, Status)

End Sub
```

```
Private Sub cExecStep73_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep74_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep74_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep74_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep74.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep74_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep75_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep75_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep75_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep75.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep75_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep76_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep76_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep76_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep76.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep76_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep77_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)
```

```
End Sub

Private Sub cExecStep77_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep77_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep77.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep77_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep78_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep78_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep78_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep78.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep78_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep79_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep79_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep79_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep79.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep79_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep80_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep80_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep80_StepComplete(cStepRecord As cStep, _
```

```
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep80.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep80_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep81_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep81_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep81_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep81.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep81_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep82_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep82_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep82_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep82.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep82_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep83_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep83_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep83_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep83.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep83_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)
```

```
End Sub

Private Sub cExecStep84_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep84_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep84_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep84.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep84_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep85_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep85_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep85_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep85.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep85_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep86_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep86_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep86_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep86.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep86_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep87_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep87_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)
```

```
    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep87_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep87.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep87_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep88_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep88_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep88_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep88.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep88_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep89_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep89_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep89_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep89.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep89_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep90_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep90_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep90_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep90.Index,
InstanceId, Status)
```

```
End Sub

Private Sub cExecStep90_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep91_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep91_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep91_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep91.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep91_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep92_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep92_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep92_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep92.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep92_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep93_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep93_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep93_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep93.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep93_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep94_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)
```

```
    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep94_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep94_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep94.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep94_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep95_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep95_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep95_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep95.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep95_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep96_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep96_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep96_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep96.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep96_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep97_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep97_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub
```

```
Private Sub cExecStep97_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep97.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep97_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep98_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep98_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep98_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep98.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep98_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep99_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep99_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep99_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep99.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep99_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep2_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep2_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep2_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep2.Index, _
            InstanceId, Status)

End Sub

Private Sub cExecStep2_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)
```

```
    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub


Private Sub cExecStep3_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep3_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep3_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep3.Index, _
            InstanceId, Status)

End Sub

Private Sub cExecStep3_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub


Private Sub cExecStep4_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep4_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep4_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep4.Index, _
            InstanceId, Status)

End Sub


Private Sub cExecStep4_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep5_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep5_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep5_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep5.Index, _
            InstanceId, Status)

End Sub

Private Sub cExecStep5_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep6_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)
```

```
    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep6_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep6_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep6.Index, _
            InstanceId, Status)

End Sub

Private Sub cExecStep6_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep7_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep7_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep7_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep7.Index, _
            InstanceId, Status)

End Sub

Private Sub cExecStep7_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep8_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep8_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long)

    Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep8_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)

    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep8.Index, _
            InstanceId, Status)

End Sub

Private Sub cExecStep8_StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)

    Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub Class_Initialize()

    Dim lngCount As Long
    Dim lngTemp As Long

    On Error GoTo InitializeErr

    Set mcFreeSteps = New cVectorLng
    ' Initialize the array of free objects with all elements
    ' for now
    For lngCount = 0 To glngNumConcurrentProcesses - 1 Step 1
        mcFreeSteps.Add lngCount
    Next lngCount

    ' Initialize a byte array with the number of free processes. It will
    ' be used later to determine if any step is running
```

```
    ' Each element in the array can represent 8 steps, 1 for each bit
    ReDim mbarrFree(glngNumConcurrentProcesses \ gintBitsPerByte)

    ' Initialize each element in the byte array w/ all 1's
    ' (upto glngNumConcurrentProcesses)
    For lngCount = LBound(mbarrFree) To UBound(mbarrFree) Step 1
        lngTemp = IIf( _
                glngNumConcurrentProcesses - (gintBitsPerByte *
lngCount) > gintBitsPerByte, _
                gintBitsPerByte, _
                glngNumConcurrentProcesses - (gintBitsPerByte *
lngCount))

        mbarrFree(lngCount) = (2 ^ lngTemp) - 1
    Next lngCount

    Set mcInstances = New cInstances
    Set mcFailures = New cFailedSteps
    Set mcNavSteps = New cStepTree
    Set mcTermSteps = New cTermSteps

    ' Initialize the Abort flag to False
    mblnAbort = False
    mblnAsk = False

    Exit Sub

InitializeErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInitializeFailed, mstrModuleName &
"Initialize", _
            LoadResString(errInitializeFailed)

End Sub
Private Sub Class_Terminate()

    On Error GoTo Class_TerminateErr

    mcFreeSteps.Clear
    Set mcFreeSteps = Nothing
    ReDim mbarrFree(0)

    mcInstances.Clear
    Set mcInstances = Nothing

    Set mcFailures = Nothing
    Set mcNavSteps = Nothing
    Set mcTermSteps = Nothing

    Exit Sub

Class_TerminateErr:
    Call LogErrors(Errors)
End Sub

Private Sub mcTermSteps_TermStepExists(cStepDetails As cTermStep)

    Call RunNextStep(cStepDetails.TimeComplete, cStepDetails.Index, _
            cStepDetails.InstanceId, cStepDetails.ExecutionStatus)

End Sub
```

cRunItDetails.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunItDetails"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cRunItDetails.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module encapsulates the properties of iterator
values
'               that are used by the step being executed at runtime.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'

Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cRunItDetails."
Private mstrSource As String

Private mstrIteratorName As String
Private mintType As ValueType
Private mlngSequence As Long
Private mlngFrom As Long
Private mlngTo As Long
Private mlngStep As Long
Private mstrValue As String

Public Property Get RangeTo() As Long

    RangeTo = mlngTo

End Property
Public Property Let RangeTo(ByVal vdata As Long)

    mlngTo = vdata
```

```
End Property

Public Property Get RangeFrom() As Long

    RangeFrom = mlngFrom

End Property
Public Property Get Sequence() As Long

    Sequence = mlngSequence

End Property

Public Property Get RangeStep() As Long

    RangeStep = mlngStep

End Property
Public Property Let RangeStep(vdata As Long)

    mlngStep = vdata

End Property

Public Property Let RangeFrom(ByVal vdata As Long)

    mlngFrom = vdata

End Property
Public Property Let Sequence(ByVal vdata As Long)

    mlngSequence = vdata

End Property

Public Property Get IteratorType() As ValueType

    IteratorType = mintType

End Property
Public Property Let IteratorType(ByVal vdata As ValueType)

    On Error GoTo TypeErr
    mstrSource = mstrModuleName & "Type"

    ' These constants have been defined in the enumeration,
    ' Type, which is exposed
    Select Case vdata
        Case gintFrom, gintTo, gintStep, gintValue
            mintType = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errTypeInvalid, _
                mstrSource, LoadResString(errTypeInvalid)
    End Select
    Exit Property

TypeErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Type"
    On Error GoTo 0
    Err.Raise vbObjectError + errTypeInvalid, _
        mstrSource, LoadResString(errTypeInvalid)

End Property
Private Sub IsList()

    If mintType <> gintValue Then
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidProperty, mstrSource, _
                LoadResString(errInvalidProperty)
    End If

End Sub
Private Sub IsRange()

    If mintType = gintValue Then
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidProperty, mstrSource, _
                LoadResString(errInvalidProperty)
    End If

End Sub

Public Property Get Value() As String

    Value = mstrValue

End Property
Public Property Let Value(vdata As String)

    mstrValue = vdata

End Property


Public Property Get IteratorName() As String

    IteratorName = mstrIteratorName

End Property
Public Property Let IteratorName(ByVal vdata As String)

    mstrIteratorName = vdata

End Property
```

cRunItNode.cls

```
VERSION 1.0 CLASS
```

```
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunItNode"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
' An iterator class containing the properties that are used
' by the stpe being executed.
' These iterators might actually come from steps that are at
' a higher level than the step actually being executed (viz.
' direct ascendants of the step at any level).

Option Explicit

Public IteratorName As String
Public Value As String
Public StepId As Long
```

cRunOnly.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunOnly"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Option Explicit

Public Event Done()
Private WithEvents mcRunWsp As cRunWorkspace
Attribute mcRunWsp.VB_VarHelpID = -1

Public WspName As String
Public WorkspaceId As Long
Public WspLog As String

Public Sub RunWsp()

    On Error GoTo RunWspErr

    Set mcRunWsp = New cRunWorkspace
    Set mcRunWsp.LoadDb = dbsAttTool
    mcRunWsp.WorkspaceId = WorkspaceId
    mcRunWsp.CreateInputFiles = True
    mcRunWsp.RunWorkspace

    Exit Sub

RunWspErr:
    ' Log the VB error code
    LogErrors Errors
End Sub

Private Sub mcRunWsp_RunComplete(dtmEndTime As Currency)

    MsgBox "Completed executing workspace: " & gstrSQ & WspName & gstrSQ
& " at " & _
            JulianDateToString(dtmEndTime) & "." & vbCrLf & vbCrLf & _
            "The log file for the run is: " & gstrSQ & WspLog & gstrSQ &
"."
    RaiseEvent Done

End Sub

Private Sub mcRunWsp_RunStart(dtmStartTime As Currency, strWspLog As
String, lRunId As Long)
    WspLog = strWspLog
End Sub
```

cRunStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cRunStep.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This class executes the step that is assigned to the
'               ExecuteStep property. It executes the pre-execution
constraints
'               in sequence and then the step itself. At the end it
executes
'               the post-execution constraints. Since these steps should
always
'               be executed in sequence, each step is only fired on the
'               completion of the previous step.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cRunStep."
Private mstrSource As String

' Local variable(s) to hold property value(s)
```

```
Private mcStep As cStep
Private mcGlobals As cArrSteps
Private mcvntWspPreCons As Variant
Private mcvntWspPostCons As Variant
Private mcvntPreCons As Variant
Private mcvntPostCons As Variant
Private mcIterators As cRunColIt
Private mlngInstanceId As Long ' Identifier for the current instance
Private mlngIndex As Long ' Index value for the current instance
Private mstrCommand As String ' The command string
Private msRunStepDtl As String ' Step text/file name that will go into
the run_step_details table
Private mblnAbort As Boolean   ' Set to True when the user aborts the
run
Private msOutputFile As String
Private msErrorFile As String
Private miStatus As InstanceStatus
Private mcVBErr As cVBErrorsSM
Public WspParameters As cArrParameters
Public WspConnections As cConnections
Public WspConnDtls As cConnDtls

Private WithEvents mcTermProcess As cTermProcess
Attribute mcTermProcess.VB_VarHelpID = -1
Public RunId As Long
Public CreateInputFiles As Boolean
Private msOutputDir As String

' Object that will execute the step
Private WithEvents mcExecObj As EXECUTEDLLLib.Execute
Attribute mcExecObj.VB_VarHelpID = -1

' Holds the step that is currently being executed (constraint or
' worker step)
Private mcExecStep As cStep

Private Const msCompareExe As String = "\diff.exe"

Private Enum NextNodeType
    mintWspPreConstraint = 1
    mintPreConstraint
    mintStep
    mintWspPostConstraint
    mintPostConstraint
End Enum

' Public events to notify the calling function of the
' start and end time for each step
Public Event StepStart(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long)
Public Event StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, InstanceId As Long, Status As
InstanceStatus)
Public Event ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, _
        InstanceId As Long)
Public Event ProcessComplete(cStepRecord As cStep, _
        dtmStartTime As Currency, InstanceId As Long, lElapsed As Long)

Private Function AppendDiffErrors(sDiffFile As String)
    ' The file containing the errors generated by the diff utility is
passed in
    ' These errors are appended to the error file for the step

    Dim sTemp As String
    Dim InputFile As Integer

    If Not StringEmpty(sDiffFile) Then

        InputFile = FreeFile
        Open sDiffFile For Input Access Read As InputFile

        Do While Not EOF(InputFile)         ' Loop until end of file.
            Line Input #InputFile, sTemp    ' Read line into variable.
            mcVBErr.LogMessage sTemp
        Loop

        Close InputFile
    End If

End Function

Private Sub CreateStepTextFile()
    ' Creates a file containing the step text being executed
    On Error GoTo CreateStepTextFileErr

    Dim sInputFile As String

    If mcExecStep.ExecutionMechanism = gintExecuteShell Then
        sInputFile = GetOutputFile(gsCmdFileSuffix)
    Else
        sInputFile = GetOutputFile(gsSqlFileSuffix)
    End If

    ' Generate a file containing the step text being executed
    If Not StringEmpty(mcExecStep.StepTextFile) Or
mcExecStep.ExecutionMechanism = gintExecuteShell Then
        FileCopy mstrCommand, sInputFile
    Else
        Call WriteCommandToFile(mstrCommand, sInputFile)
    End If

    Exit Sub

CreateStepTextFileErr:

    mcVBErr.LogVBErrors

End Sub
Private Function GetOutputFile(strFileExt As String) As String
    ' This function generates the output file name for the step
currently being executed
```

```
    ' The value of the built-in parameter 'DefaultDir' is appended with
the run identifier
    ' for the file location
    ' The step label is used for the file name and a combination of all
iterator values
    ' for the step is used to make the output files unique for each
instance
    Dim sFile As String
    Dim sIt As String
    Dim lIt As Long

    On Error GoTo GetOutputFileErr

    sFile = SubstituteParametersIfPossible(mcExecStep.StepLabel)

    sFile = TranslateStepLabel(sFile)

    If mcExecStep Is mcStep Then
        ' Use iterators that have been defined for the worker or any of
it's managers
        ' to make the error/log file unique for this instance
        For lIt = mcIterators.Count - 1 To 0 Step -1
            sIt = sIt & gsExtSeparator & mcIterators(lIt).Value
        Next lIt
    End If
    sIt = sIt & strFileExt

    ' Ensure that the length of the complete path does not exceed 255
characters
    If Len(msOutputDir) + Len(sFile) + Len(sIt) > MAX_PATH Then
        sFile = Mid(sFile, 1, MAX_PATH - Len(sIt) - Len(msOutputDir))
    End If
    GetOutputFile = msOutputDir & sFile & sIt
    Exit Function

GetOutputFileErr:

    ' Does not make sense to log error to the error file yet. Write to
the project
    ' log and return the step label as default
    GetOutputFile = mcExecStep.StepLabel & gsExtSeparator & strFileExt

End Function


Private Sub HandleExecutionError()

    On Error GoTo HandleExecutionError

    ' Log the error code raised by Visual Basic
    miStatus = gintFailed
    mcVBErr.LogVBErrors
    Call mcVBErr.WriteError(errExecuteStepFailed, _
            OptArgs:="Continuation criteria for the step is: " &
gsContCriteria(mcStep.ContinuationCriteria))

HandleExecutionError:

    ' Logging failed - return

End Sub

Public Property Get Index() As Long

    Index = mlngIndex

End Property
Public Property Let Index(ByVal vdata As Long)

    mlngIndex = vdata

End Property
Private Function InitializeExecStatus() As InstanceStatus
    Dim sCompareFile As String

    On Error GoTo InitializeExecStatusErr

    InitializeExecStatus = mcExecObj.StepStatus

    If InitializeExecStatus = gintComplete Then
        If Not StringEmpty(mcExecStep.FailureDetails) Then
            ' Compare output to determine whether the step failed
            sCompareFile = GetShortName(SubstituteParameters( _
                    mcExecStep.FailureDetails, mcExecStep.WorkspaceId,
mcIterators, _
                    WspParameters))
            InitializeExecStatus = IIf(CompareOutput(sCompareFile,
msOutputFile), gintComplete, gintFailed)
        End If
    End If

    Exit Function

InitializeExecStatusErr:
    mcVBErr.LogVBErrors
'    Call LogErrors(Errors)
    InitializeExecStatus = mcExecObj.StepStatus

End Function
Private Function CompareOutput(sCompareFile As String, sOutputFile As
String) As Boolean

    Dim sCmpOutput As String
    Dim sDiffOutput As String

    On Error GoTo CompareOutputErr

    ' Create temporary files to store the file compare output and
    ' the errors generated by the compare function
    sCmpOutput = CreateTempFile()
    sDiffOutput = CreateTempFile()
```

```
    ' Run the compare utility and redirect it's output and errors
    SyncShell ("cmd /c " & _
            GetShortName(App.Path & msCompareExe) & gstrBlank & _
            sCompareFile & gstrBlank & sOutputFile & _
            " > " & sCmpOutput & " 2> " & sDiffOutput)

    If FileLen(sDiffOutput) > 0 Then
        ' The compare generated errors - append error msgs to the error
file
        Call AppendDiffErrors(sDiffOutput)
        CompareOutput = False
    Else
        CompareOutput = (FileLen(sCmpOutput) = 0)
    End If

    If Not CompareOutput Then
        mcVBErr.WriteError errDiffFailed
    End If

    ' Delete the temporary files used to store the output of the compare
and
    ' the errors generated by the compare
    Kill sDiffOutput
    Kill sCmpOutput

    Exit Function

CompareOutputErr:
    mcVBErr.LogVBErrors
    CompareOutput = False

End Function
Public Property Get InstanceId() As Long

    InstanceId = mlngInstanceId

End Property
Public Property Let InstanceId(ByVal vdata As Long)

    mlngInstanceId = vdata

End Property

Private Function ExecuteConstraint(vntConstraints As Variant, _
        ByRef intLoopIndex As Integer) As Boolean

    ' Returns True if there is a constraint in the passed in
    ' array that remains to be executed

    If IsArray(vntConstraints) And Not IsEmpty(vntConstraints) Then
        ExecuteConstraint = (LBound(vntConstraints) <= intLoopIndex) And
(intLoopIndex <= UBound(vntConstraints))
    Else
        ExecuteConstraint = False
    End If
End Function
Private Function NextStep() As cStep

    ' Determines which is the next step to be executed - it could
    ' be either a pre-execution step, the worker step itself
    ' or a post-execution step

    Dim cConsRec As cConstraint
    Dim cNextStepRec As cStep
    Dim vntStepConstraints As Variant

    ' Static variable to remember exactly where we are in the
    ' processing
    Static intIndex As Integer
    Static intNextStepType As NextNodeType

    On Error GoTo NextStepErr

    If mblnAbort = True Then
        ' The user has aborted the run - do not run any more
        ' processes for the step
        Set NextStep = Nothing
        Exit Function
    End If

    If intNextStepType = 0 Then
        ' First time through this function - set the Index and
        ' node type to initial values
        intNextStepType = mintWspPreConstraint
        intIndex = 0
        RaiseEvent StepStart(mcStep, Determine64BitTime(),
mlngInstanceId)
    End If

    Do
        Select Case intNextStepType
            Case mintWspPreConstraint
                vntStepConstraints = mcvntWspPreCons

            Case mintPreConstraint
                vntStepConstraints = mcvntPreCons

            Case mintStep
                ' CONS:
                If mcStep.StepType = gintWorkerStep Then
                    Set cNextStepRec = mcStep
                End If

            Case mintWspPostConstraint
                vntStepConstraints = mcvntWspPostCons

            Case mintPostConstraint
                vntStepConstraints = mcvntPostCons

        End Select
```

```
        If intNextStepType <> mintStep Then
            ' Check if there is a constraint to be executed
            If ExecuteConstraint(vntStepConstraints, intIndex) Then
                ' Get the corresponding step record to be executed
                ' Query the global step record for the current
                ' constraint
                Set cConsRec = vntStepConstraints(intIndex)

                Set cNextStepRec =
mcGlobals.QueryStep(cConsRec.GlobalStepId)
                intIndex = intIndex + 1
            Else
                If intNextStepType = mintPostConstraint Then
                    ' No more stuff to be executed for the step
                    ' Raise a Done event
                    Set cNextStepRec = Nothing

                    ' Set the next step type to an invalid value
                    intNextStepType = -1
                Else
                    Call NextType(intNextStepType, intIndex)
                End If
            End If
        Else
            ' Increment the step type so we look at the post-
            ' execution steps the next time through
            Call NextType(intNextStepType, intIndex)
        End If

    Loop Until (Not cNextStepRec Is Nothing) Or _
            intNextStepType = -1

    Set NextStep = cNextStepRec

    Exit Function

NextStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "NextStep"
    Err.Raise vbObjectError + errNextStepFailed, mstrSource, _
        LoadResString(errNextStepFailed)

End Function
Public Sub Execute()
    ' This procedure is the method that executes the step that
    ' is assigned to the ExecuteStep property. It call a procedure
    ' to determine the next step to be executed.
    ' Then it initializes all the properties of the cExecuteSM object
    ' and calls it's run method to execute it.
    Dim cConn As cConnection
    Dim cRunConnDtl As cConnDtl

    On Error GoTo ExecuteErr

    ' If this procedure is called after a step has completed,
    ' we would have to check if we created any temporary files
    ' while executing that step
    If Not mcExecStep Is Nothing Then
        If Not StringEmpty(mcExecStep.StepTextFile) Or
mcExecStep.ExecutionMechanism = gintExecuteShell Then
            ' Remove the temporary file that we created while
            ' running this command
            Kill mstrCommand
        End If

        Call StepCompleted

        ' The VB errors class stores a reference to the Execute class
since it uses
        ' a method of the class to write errors to the error log. Hence,
        ' release all references to the Execute object before destroying
it.
        Set mcVBErr.ErrorFile = Nothing
        Set mcExecObj = Nothing

        ' Delete empty output and error files (generated by shell
commands)
        ' (Can be done only after cleaning up cExecObj)
        Call DeleteEmptyOutputFiles
    Else
        ' First time through - initialize the location of output files
        msOutputDir = GetDefaultDir(mcStep.WorkspaceId, WspParameters)
        msOutputDir = msOutputDir & gstrFileSeparator & Trim(Str(RunId))
& gstrFileSeparator
        ' Dummy file since the function expects a file name
        MakePathValid (msOutputDir & "a.txt")
    End If

    ' Call a procedure to determine the next step to be executed
    ' - could be a constraint or the step itself
    ' Initialize a module-level variable to the step being
    ' executed
    Set mcExecStep = NextStep
    If mcExecStep Is Nothing Then
        RaiseEvent StepComplete(mcStep, Determine64BitTime(),
mlngInstanceId, miStatus)
        ' No more stuff to execute
        Exit Sub
    End If

    Dim sStartDir As String

    Set mcExecObj = New EXECUTEDLLLib.Execute

    ' The VB errors class uses the WriteError method of the Execute
class to write
    ' all VB errors to the error file for the step (this prevents a
clash when the
    ' VB errors and Execution errors have to be written to the same log).
Hence, store
```

```
    ' a reference to the Execute object in mcVBErr
    msErrorFile = GetOutputFile(gsErrorFileSuffix)
    mcExecObj.ErrorFile = msErrorFile
    Call DeleteFile(msErrorFile, bCheckIfEmpty:=False)
    Set mcVBErr.ErrorFile = mcExecObj

    If mcExecStep.ExecutionMechanism = gintExecuteShell Then
        sStartDir = Trim$(GetShortName(SubstituteParameters( _
                    mcExecStep.StartDir, mcExecStep.WorkspaceId,
mcIterators, WspParameters:=WspParameters)))
        ' Dummy connection object
        Set cConn = New cConnection
        Set cRunConnDtl = New cConnDtl
    Else
        ' Find the connection string value and substitute parameter
values in it
        Set cRunConnDtl =
WspConnDtls.GetConnectionDtl(mcExecStep.WorkspaceId,
mcExecStep.StartDir)
        Set cConn = WspConnections.GetConnection(mcExecStep.WorkspaceId,
cRunConnDtl.ConnectionString)
        sStartDir = Trim$(SubstituteParameters(cConn.ConnectionValue, _
                mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters))
    End If

    msOutputFile = GetOutputFile(gsOutputFileSuffix)
    Call DeleteFile(msOutputFile, bCheckIfEmpty:=False)
    mcExecObj.OutputFile = msOutputFile
'    mcExecObj.LogFile = GetShortName(SubstituteParameters( _
'            mcExecStep.LogFile, mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters))
    If mcExecStep.ExecutionMechanism = gintExecuteODBC And _
            cRunConnDtl.ConnType = ConnTypeDynamic Then
        Call mcExecObj.DoExecute(BuildCommandString(), sStartDir,
mcExecStep.ExecutionMechanism, _
                cConn.NoCountDisplay, cConn.NoExecute,
cConn.ParseQueryOnly, cConn.QuotedIdentifiers, _
                cConn.AnsiNulls, cConn.ShowQueryPlan,
cConn.ShowStatsTime, cConn.ShowStatsIO, _
                cConn.RowCount, cConn.QueryTimeOut, gstrEmptyString)
    Else
        Call mcExecObj.DoExecute(BuildCommandString(), sStartDir,
mcExecStep.ExecutionMechanism, _
                cConn.NoCountDisplay, cConn.NoExecute,
cConn.ParseQueryOnly, cConn.QuotedIdentifiers, _
                cConn.AnsiNulls, cConn.ShowQueryPlan,
cConn.ShowStatsTime, cConn.ShowStatsIO, _
                cConn.RowCount, cConn.QueryTimeOut, mcExecStep.StartDir)
    End If

    Exit Sub

ExecuteErr:
    Call HandleExecutionError
    ' We can assume that if we are in this function, a StepStart event
has been triggered already.
    RaiseEvent StepComplete(mcStep, Determine64BitTime(), mlngInstanceId,
miStatus)

End Sub
Private Function BuildCommandString() As String
    ' Process text to be executed - either from the text
    ' field or read it from a file.
    ' This function will always return the command text for ODBC
commands
    ' and a file name for Shell commands
    Dim sFile As String
    Dim sCommand As String
    Dim sTemp As String

    On Error GoTo BuildCommandStringErr

    If Not StringEmpty(mcExecStep.StepTextFile) Then
        ' Substitute parameter values and environment variables
        ' in the filename
        msRunStepDtl = SubstituteParameters(mcExecStep.StepTextFile, _
                mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters)

        sFile = GetShortName(msRunStepDtl)

        mstrCommand = SubstituteParametersInText(sFile,
mcExecStep.WorkspaceId)

        If mcExecStep.ExecutionMechanism = gintExecuteODBC Then
            ' Read the contents of the file and pass it to ODBC
            BuildCommandString = ReadCommandFromFile(mstrCommand)
        Else
            BuildCommandString = mstrCommand
        End If
    Else
        ' Substitute parameter values and environment variables
        ' in the step text
        msRunStepDtl = SubstituteParameters(mcExecStep.StepText, _
                mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters)
        mstrCommand = msRunStepDtl

        If mcExecStep.ExecutionMechanism = gintExecuteShell Then
            ' Write the command to a temp file (enables us to execute
multiple
            ' commands via the command interpreter)
            mstrCommand = WriteCommandToFile(msRunStepDtl)
            BuildCommandString = mstrCommand
        Else
            BuildCommandString = SQLFixup(msRunStepDtl)
        End If
    End If

    If CreateInputFiles Then
```

```
        Call CreateStepTextFile
    End If

    Exit Function

BuildCommandStringErr:
    ' Log the error code raised by the Execute procedure
'    Call LogErrors(Errors)
    mcVBErr.LogVBErrors

    On Error GoTo 0
    mstrSource = mstrModuleName & "Execute"
    Err.Raise vbObjectError + errExecuteStepFailed, mstrSource, _
        LoadResString(errExecuteStepFailed) & mstrCommand

End Function
Public Sub Abort()

    On Error GoTo AbortErr

    ' Setting the Abort flag to True will ensure that we
    ' don't execute any more processes for this step
    mblnAbort = True

    If Not mcExecObj Is Nothing Then
        mcExecObj.Abort
    Else
        ' We are not in the middle of execution yet
    End If

    Exit Sub

AbortErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, _
        mstrModuleName & "Abort", _
        LoadResString(errProgramError)

End Sub
Private Sub NextType(ByRef StepType As NextNodeType, _
        ByRef Position As Integer)

    StepType = StepType + 1
    Position = 0

End Sub
Private Sub StepCompleted()

    On Error GoTo StepCompletedErr

    If Not mcExecStep Is Nothing Then
        If mcExecStep Is mcStep Then
            miStatus = InitializeExecStatus
            If miStatus = gintFailed Then
                ' Create input files if the step failed execution and
one hasn't been created already
                If Not CreateInputFiles Then CreateStepTextFile
                Call mcVBErr.WriteError(errExecuteStepFailed, _
                        OptArgs:="Continuation criteria for the step is:
" & gsContCriteria(mcStep.ContinuationCriteria))
            End If
        End If
    End If

    Exit Sub

StepCompletedErr:
    ' Log the error code raised by Visual Basic
    miStatus = gintFailed
    mcVBErr.LogVBErrors
    Call mcVBErr.WriteError(errExecuteStepFailed, _
            OptArgs:="Continuation criteria for the step is: " &
gsContCriteria(mcStep.ContinuationCriteria))

End Sub
Private Sub DeleteEmptyOutputFiles()

    On Error GoTo DeleteEmptyOutputFilesErr

    ' Delete empty output and error files
    If Not mcExecStep Is Nothing Then
        Call DeleteFile(msErrorFile, bCheckIfEmpty:=True)
        Call DeleteFile(msOutputFile, bCheckIfEmpty:=True)
    End If

    Exit Sub

DeleteEmptyOutputFilesErr:
    ' Not a critical error - continue

End Sub
Private Function ReadCommandFromFile(strFileName As String) As String

    ' Returns the contents of the passed in file

    Dim sCommand As String
    Dim sTemp As String
    Dim InputFile As Integer

    On Error GoTo ReadCommandFromFileErr

    If Not StringEmpty(strFileName) Then

        InputFile = FreeFile
        Open strFileName For Input Access Read As InputFile

        Line Input #InputFile, sCommand ' Read line into variable.

        Do While Not EOF(InputFile) ' Loop until end of file.
            Line Input #InputFile, sTemp ' Read line into variable.
```

```
            sCommand = sCommand & vbCrLf & sTemp
        Loop

        Close InputFile
    End If

    ReadCommandFromFile = sCommand

    Exit Function

ReadCommandFromFileErr:

    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "ReadCommandFromFile"
    On Error GoTo 0
    Err.Raise vbObjectError + errSubValuesFailed, _
            gstrSource, _
            LoadResString(errSubValuesFailed)

End Function
Private Function SubstituteParametersIfPossible(strLabel As String)

    On Error GoTo SubstituteParametersIfPossibleErr

    SubstituteParametersIfPossible = SubstituteParameters(strLabel, _
            mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters)
    Exit Function

SubstituteParametersIfPossibleErr:
    SubstituteParametersIfPossible = strLabel

End Function
Private Function SubstituteParametersInText(strFileName As String, _
        lngWorkspace As Long) As String

    ' Reads each line in the passed in file, substitutes parameter
    ' values in the line and writes out the modified line to a
    ' temporary file that we create. The temporary file will be
    ' removed once the step completes execution.
    ' Returns the name of the newly created temporary file.

    Dim strTempFile As String
    Dim strTemp As String
    Dim strOutput As String
    Dim InputFile As Integer
    Dim OutputFile As Integer

    On Error GoTo SubstituteParametersInTextErr

    strTempFile = CreateTempFile()

    If Not StringEmpty(strFileName) Then

        InputFile = FreeFile
        Open strFileName For Input Access Read As InputFile

        OutputFile = FreeFile
        Open strTempFile For Output Access Write As OutputFile

        Do While Not EOF(InputFile) ' Loop until end of file.
            Line Input #InputFile, strTemp ' Read line into variable.
            strOutput = SubstituteParameters(strTemp, lngWorkspace,
mcIterators, WspParameters:=WspParameters)

            If mcExecStep.ExecutionMechanism = gintExecuteODBC Then
strOutput = SQLFixup(strOutput)

            Print #OutputFile, strOutput
            BugMessage strOutput
        Loop

    End If

    Close InputFile
    Close OutputFile

    SubstituteParametersInText = strTempFile

    Exit Function

SubstituteParametersInTextErr:

    ' Log the error code raised by Visual Basic
'    Call LogErrors(Errors)
    mcVBErr.LogVBErrors
    mstrSource = mstrModuleName & "SubstituteParametersInText"
    On Error GoTo 0
    Err.Raise vbObjectError + errSubValuesFailed, _
            gstrSource, _
            LoadResString(errSubValuesFailed)

End Function

Private Function WriteCommandToFile(sCommand As String, Optional sFile
As String = gstrEmptyString) As String

    ' Writes the command text to a temporary file
    ' Returns the name of the temporary file

    Dim OutputFile As Integer

    On Error GoTo WriteCommandToFileErr

    If StringEmpty(sFile) Then
        sFile = CreateTempFile()
    End If

    OutputFile = FreeFile
    Open sFile For Output Access Write As OutputFile
```

```
    Print #OutputFile, sCommand

    Close OutputFile

    WriteCommandToFile = sFile

    Exit Function

WriteCommandToFileErr:

    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "WriteCommandToFile"
    On Error GoTo 0
    Err.Raise vbObjectError + errSubValuesFailed, _
            gstrSource, _
            LoadResString(errSubValuesFailed)

End Function


Public Property Get WspPreCons() As Variant
    WspPreCons = mcvntWspPreCons
End Property
Public Property Let WspPreCons(ByVal vdata As Variant)
    mcvntWspPreCons = vdata
End Property

Public Property Get WspPostCons() As Variant
    WspPostCons = mcvntWspPostCons
End Property
Public Property Let WspPostCons(ByVal vdata As Variant)
    mcvntWspPostCons = vdata
End Property

Public Property Get PreCons() As Variant
    PreCons = mcvntPreCons
End Property
Public Property Let PreCons(ByVal vdata As Variant)
    mcvntPreCons = vdata
End Property

Public Property Get PostCons() As Variant
    PostCons = mcvntPostCons
End Property
Public Property Let PostCons(ByVal vdata As Variant)
    mcvntPostCons = vdata
End Property

Public Property Set Globals(cRunSteps As cArrSteps)

    Set mcGlobals = cRunSteps

End Property
Public Property Set ExecuteStep(cRunStep As cStep)

    Set mcStep = cRunStep

End Property
Public Property Get Globals() As cArrSteps

    Set Globals = mcGlobals

End Property
Public Property Get ExecuteStep() As cStep

    Set ExecuteStep = mcStep

End Property
Public Property Set Iterators(vdata As cRunColIt)

    Set mcIterators = vdata

End Property
Private Sub Class_Initialize()

    ' Initialize the Abort flag to False
    mblnAbort = False
    Set mcVBErr = New cVBErrorsSM
    Set mcTermProcess = New cTermProcess

End Sub

Private Sub Class_Terminate()

    On Error GoTo Class_TerminateErr

    Set mcExecObj = Nothing
    Set mcVBErr = Nothing
    Set mcTermProcess = Nothing

    Exit Sub

Class_TerminateErr:
    Call LogErrors(Errors)

End Sub

Private Sub mcExecObj_Start(ByVal StartTime As Currency)
    ' Raise an event indicating that the step has begun execution
    RaiseEvent ProcessStart(mcExecStep, msRunStepDtl, StartTime,
mlngInstanceId)
End Sub

Private Sub mcExecObj_Complete(ByVal EndTime As Currency, ByVal Elapsed
As Long)

    On Error GoTo mcExecObj_CompleteErr

    Debug.Print Elapsed
    RaiseEvent ProcessComplete(mcExecStep, EndTime, mlngInstanceId,
Elapsed)
```

```
    mcTermProcess.ProcessTerminated

    Exit Sub

mcExecObj_CompleteErr:
    Call LogErrors(Errors)

End Sub

Private Sub mcTermProcess_TermProcessExists()

    On Error GoTo TermProcessExistsErr

    ' Call a procedure to execute the next step, if any
    Call Execute

    Exit Sub

TermProcessExistsErr:
    ' Log the error code raised by the Execute procedure
    Call LogErrors(Errors)

End Sub
```

cRunWorkspace.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunWorkspace"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cRunWorkspace.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This class loads all the information necessary to
'               execute a workspace and calls cRunInst to execute the
workspace.
'               It also propagates Step start and complete and
'               Run start and complete events.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "cRunWorkspace."
Private mstrSource As String

Private mcRunSteps As cArrSteps
Private mcRunParams As cArrParameters
Private mcRunConstraints As cArrConstraints
Private mcRunConnections As cConnections
Private mcRunConnDtls As cConnDtls
Private mcvntWspPreCons As Variant
Private mcvntWspPostCons As Variant
Private mdbsLoadDb As Database
Private mlngRunId As Long
Private mlngWorkspaceId As Long
Private mField As cStringSM
Public CreateInputFiles As Boolean

Private WithEvents mcRun As cRunInst
Attribute mcRun.VB_VarHelpID = -1

Public Event RunStart(dtmStartTime As Currency, strWspLog As String,
lRunId As Long)
Public Event RunComplete(dtmEndTime As Currency)
Public Event StepStart(cStepRecord As cStep, dtmStartTime As Currency,
lngInstanceId As Long, _
        sPath As String, sIts As String)
Public Event StepComplete(cStepRecord As cStep, dtmEndTime As Currency,
lngInstanceId As Long)
Public Event ProcessStart(cStepRecord As cStep, strCommand As String, _
        dtmStartTime As Currency, lngInstanceId As Long)
Public Event ProcessComplete(cStepRecord As cStep, dtmEndTime As
Currency, lngInstanceId As Long)
Public Function InstancesForStep(lngStepId As Long, iStatus As
InstanceStatus) As cInstances
    ' Returns an array of all the instances for a step

    If mcRun Is Nothing Then
        Set InstancesForStep = Nothing
    Else
        Set InstancesForStep = mcRun.InstancesForStep(lngStepId,
iStatus)
    End If

End Function
Private Sub InsertRunDetail(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, _
        lngInstanceId As Long, lParentInstanceId As Long, sItValue As
String)
    ' Inserts a new run detail record into the database

    Dim strInsert As String
    Dim qy As QueryDef

    On Error GoTo InsertRunDetailErr
    mstrSource = mstrModuleName & "InsertRunDetail"

    strInsert = "insert into run_step_details " & _
            "( run_id, step_id, version_no, instance_id,
parent_instance_id, " & _
            " command, start_time, iterator_value ) " & _
            " values ( "
```

```
#If USE_JET Then

    strInsert = strInsert & " [r_id], [s_id], [ver_no], [i_id], [p_i_id],
" & _
            " [com], [s_date], [it_val] )"

    Set qy = mdbsLoadDb.CreateQueryDef( _
            gstrEmptyString, strInsert)

    ' Call a procedure to assign the Querydef parameters
    Call AssignParameters(qy, StartTime:=dtmStartTime, _
            StepId:=cStepRecord.StepId, _
            Version:=cStepRecord.VersionNo, _
            InstanceId:=lngInstanceId, _
            Command:=strCommand)

    qy.Execute dbFailOnError
    qy.Close

#Else

    strInsert = strInsert & Str(mlngRunId) _
        & ", " & Str(cStepRecord.StepId) _
        & ", " & mField.MakeStringFieldValid(cStepRecord.VersionNo) _
        & ", " & Str(lngInstanceId) _
        & ", " & Str(lParentInstanceId) _
        & ", " & mField.MakeStringFieldValid(strCommand) _
        & ", " & Str(dtmStartTime) _
        & ", " & mField.MakeStringFieldValid(sItValue)

    strInsert = strInsert & " ) "

    mdbsLoadDb.Execute strInsert, dbFailOnError

#End If

    Exit Sub

InsertRunDetailErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "InsertRunDetail"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateRunDataFailed, _
            mstrSource, _
            LoadResString(errUpdateRunDataFailed)

End Sub
Private Sub UpdateRunDetail(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)
    ' Updates the run detail record in the database

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo UpdateRunDetailErr

    strUpdate = "update run_step_details " & _
            " set end_time = [e_date], elapsed_time = [elapsed] " & _
            " where run_id = [r_id] " & _
            " and step_id = [s_id] " & _
            " and version_no = [ver_no] " & _
            " and instance_id = [i_id] "

    Set qy = mdbsLoadDb.CreateQueryDef( _
            gstrEmptyString, strUpdate)

    ' Call a procedure to assign the Querydef parameters
    Call AssignParameters(qy, EndTime:=dtmEndTime, _
            StepId:=cStepRecord.StepId, _
            Version:=cStepRecord.VersionNo, _
            InstanceId:=lngInstanceId, Elapsed:=lElapsed)

    qy.Execute dbFailOnError
    qy.Close

    Exit Sub

UpdateRunDetailErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "UpdateRunDetail"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateRunDataFailed, _
            mstrSource, _
            LoadResString(errUpdateRunDataFailed)

End Sub
Private Function InsertRunHeader(dtmStartTime As Currency) As Long
    ' Inserts a new run header record into the database
    ' and returns the id for the run

    Dim strInsert As String
    Dim qy As QueryDef

    On Error GoTo InsertRunHeaderErr

    strInsert = "insert into run_header " & _
        "( run_id, workspace_id, start_time ) " & _
        " values ( " & _
        " [r_id], [w_id], [s_date] )"

    Set qy = mdbsLoadDb.CreateQueryDef( _
            gstrEmptyString, strInsert)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy, StartTime:=dtmStartTime)

    qy.Execute dbFailOnError
    qy.Close

    InsertRunHeader = mlngRunId
    Exit Function
```

```
InsertRunHeaderErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "InsertRunHeader"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateRunDataFailed, _
            mstrSource, _
            LoadResString(errUpdateRunDataFailed)

End Function
Private Sub InsertRunParameters(dtmStartTime As Currency)
    ' Inserts a new run header record into the database
    ' and returns the id for the run

    Dim strInsert As String
    Dim qy As QueryDef
    Dim cParamRec As cParameter
    Dim lngIndex As Long

    On Error GoTo InsertRunParametersErr

    strInsert = "insert into run_parameters " & _
        "( run_id, parameter_name, parameter_value ) " & _
        " values ( " & _
        " [r_id], [p_name], [p_value] )"

    Set qy = mdbsLoadDb.CreateQueryDef( _
            gstrEmptyString, strInsert)
    qy.Parameters("r_id").Value = mlngRunId

    For lngIndex = 0 To mcRunParams.ParameterCount - 1
        Set cParamRec = mcRunParams(lngIndex)

        qy.Parameters("p_name").Value = cParamRec.ParameterName
        qy.Parameters("p_value").Value = cParamRec.ParameterValue
        qy.Execute dbFailOnError

    Next lngIndex

    qy.Close

    Exit Sub

InsertRunParametersErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "InsertRunParameters"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateRunDataFailed, _
            mstrSource, _
            LoadResString(errUpdateRunDataFailed)

End Sub
Private Sub AssignParameters(qyExec As DAO.QueryDef, _
        Optional StartTime As Currency = 0, _
        Optional EndTime As Currency = 0, _
        Optional StepId As Long = 0, _
        Optional Version As String = gstrEmptyString, _
        Optional InstanceId As Long = 0, _
        Optional ParentInstanceId As Long = 0, _
        Optional Command As String = gstrEmptyString, _
        Optional Elapsed As Long = 0, _
        Optional ItValue As String = gstrEmptyString)
' Assigns values to the parameters in the querydef object

Dim prmParam As DAO.Parameter

On Error GoTo AssignParametersErr
mstrSource = mstrModuleName & "AssignParameters"

For Each prmParam In qyExec.Parameters
    Select Case prmParam.Name
        Case "[w_id]"
            prmParam.Value = mlngWorkspaceId

        Case "[r_id]"
            prmParam.Value = mlngRunId

        Case "[s_id]"
            BugAssert StepId <> 0
            prmParam.Value = StepId

        Case "[ver_no]"
            BugAssert Not StringEmpty(Version)
            prmParam.Value = Version

        Case "[i_id]"
            BugAssert InstanceId <> 0
            prmParam.Value = InstanceId

        Case "[p_i_id]"
            prmParam.Value = ParentInstanceId

        Case "[com]"
            BugAssert Not StringEmpty(Command)
            prmParam.Value = Command

        Case "[s_date]"
            BugAssert StartTime <> 0
            prmParam.Value = StartTime

        Case "[e_date]"
            BugAssert EndTime <> 0
            prmParam.Value = EndTime

        Case "[elapsed]"
            prmParam.Value = Elapsed

        Case "[it_val]"
            prmParam.Value = ItValue

        Case Else
            ' Write the parameter name that is faulty
```

```
                WriteError errInvalidParameter, mstrSource, _
                        prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrSource, _
                        LoadResString(errInvalidParameter)
            End Select
        Next prmParam

        Exit Sub

AssignParametersErr:

        mstrSource = mstrModuleName & "AssignParameters"
        Call LogErrors(Errors)
        On Error GoTo 0
        Err.Raise vbObjectError + errAssignParametersFailed, _
                mstrSource, LoadResString(errAssignParametersFailed)

End Sub
Private Sub RunStartProcessing(dtmStartTime As Currency)

        On Error GoTo RunStartProcessingErr

        ' Insert the run header into the database
        Call InsertRunHeader(dtmStartTime)

        ' Insert the run parameters into the database
        Call InsertRunParameters(dtmStartTime)

        Exit Sub

RunStartProcessingErr:
        ' Log the error code raised by Visual Basic
        Call LogErrors(Errors)
        mstrSource = mstrModuleName & "RunStartProcessing"
        ShowError errUpdateRunDataFailed
        WriteError errUpdateRunDataFailed, mstrSource

End Sub
Private Sub ProcessStartProcessing(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long, _
        lParentInstanceId As Long, sItValue As String)

        On Error GoTo ProcessStartProcessingErr

        ' Insert the run detail into the database
        Call InsertRunDetail(cStepRecord, strCommand, dtmStartTime,
lngInstanceId, _
                lParentInstanceId, sItValue)

        Exit Sub

ProcessStartProcessingErr:
        ' Log the error code raised by Visual Basic
        Call LogErrors(Errors)
        mstrSource = mstrModuleName & "ProcessStartProcessing"
        ShowError errUpdateRunDataFailed
        WriteError errUpdateRunDataFailed, mstrSource

End Sub
Private Sub StepStartProcessing(cStepRecord As cStep, dtmStartTime As
Currency, _
        lngInstanceId As Long, lParentInstanceId As Long, sItValue As
String)

        On Error GoTo StepStartProcessingErr

        ' Since ProcessStart events won't be triggered for manager steps
        If cStepRecord.StepType = gintManagerStep Then
            ' Insert the run detail into the database
            Call InsertRunDetail(cStepRecord, cStepRecord.StepLabel, _
                    dtmStartTime, lngInstanceId, lParentInstanceId,
sItValue)
        End If

        Exit Sub

StepStartProcessingErr:
        ' Log the error code raised by Visual Basic
        Call LogErrors(Errors)
        mstrSource = mstrModuleName & "StepStartProcessing"
        ShowError errUpdateRunDataFailed

End Sub
Private Sub ProcessCompleteProcessing(cStepRecord As cStep, _
        dtmStartTime As Currency, lngInstanceId As Long, lElapsed As
Long)

        On Error GoTo ProcessCompleteProcessingErr

        ' Insert the run detail into the database
        Call UpdateRunDetail(cStepRecord, dtmStartTime, lngInstanceId,
lElapsed)

        Exit Sub

ProcessCompleteProcessingErr:
        ' Log the error code raised by Visual Basic
        Call LogErrors(Errors)
        mstrSource = mstrModuleName & "ProcessCompleteProcessing"
        ShowError errUpdateRunDataFailed

End Sub
Private Sub StepCompleteProcessing(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

        On Error GoTo StepCompleteProcessingErr

        ' Since ProcessComplete events won't be triggered for manager steps
        If cStepRecord.StepType = gintManagerStep Then
            ' Update the run detail in the database
```

```
        Call UpdateRunDetail(cStepRecord, dtmEndTime, lngInstanceId,
lElapsed)
    End If

    Exit Sub

StepCompleteProcessingErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ShowError errUpdateRunDataFailed

End Sub
Private Sub RunCompleteProcessing(dtmEndTime As Currency)

    On Error GoTo RunCompleteProcessingErr

    ' Update the header record with the end time for the run
    Call UpdateRunHeader(dtmEndTime)

    Exit Sub

RunCompleteProcessingErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ShowError errUpdateRunDataFailed

End Sub

Private Sub UpdateRunHeader(ByVal dtmEndTime As Currency)
    ' Updates the run header record with the end date

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo UpdateRunHeaderErr

    strUpdate = "update run_header " & _
            " set end_time = [e_date] " & _
            " where run_id = [r_id] "

    Set qy = mdbsLoadDb.CreateQueryDef( _
            gstrEmptyString, strUpdate)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy, EndTime:=dtmEndTime)

    qy.Execute dbFailOnError
    qy.Close

    Exit Sub

UpdateRunHeaderErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "UpdateRunHeader"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateRunDataFailed, _
            mstrSource, _
            LoadResString(errUpdateRunDataFailed)

End Sub

Public Property Let WorkspaceId(ByVal vdata As Long)
    mlngWorkspaceId = vdata
End Property
Public Property Get WorkspaceId() As Long
    WorkspaceId = mlngWorkspaceId
End Property
Public Sub RunWorkspace()

    Dim cRunSeq As cSequence

    On Error GoTo RunWorkspaceErr

    ' Call a procedure to load the module-level structures
    ' with all the step and parameter data for the run
    If LoadRunData = False Then
        ' Error handled by the function already
        Exit Sub
    End If

    ' Retrieve the next run identifier using the sequence class
    Set cRunSeq = New cSequence
    Set cRunSeq.IdDatabase = dbsAttTool
    cRunSeq.IdentifierColumn = "run_id"
    mlngRunId = cRunSeq.Identifier
    Set cRunSeq = Nothing

    Call mcRunParams.InitBuiltInsForRun(mlngWorkspaceId, mlngRunId)

    Set mcRun.Constraints = mcRunConstraints
    mcRun.WspPreExecution = mcvntWspPreCons
    mcRun.WspPostExecution = mcvntWspPostCons

    Set mcRun.Steps = mcRunSteps
    Set mcRun.Parameters = mcRunParams
    Set mcRun.RunConnections = mcRunConnections
    Set mcRun.RunConnDtls = mcRunConnDtls

    mcRun.WspId = mlngWorkspaceId
    mcRun.RootKey = LabelStep(mlngWorkspaceId)
    mcRun.RunId = mlngRunId
    mcRun.CreateInputFiles = CreateInputFiles

    mcRun.Run

    Exit Sub

RunWorkspaceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)

End Sub
```

```
Public Property Get LoadDb() As Database

    Set LoadDb = mdbsLoadDb

End Property
Public Property Set LoadDb(vdata As Database)

    Set mdbsLoadDb = vdata

End Property
Private Function LoadRunData() As Boolean

    ' Loads the step, parameter and constraint arrays
    ' with all the data for the workspace. Returns False
    ' if a failure occurs

    Dim strWorkspaceName As String
    Dim recWspSteps As Recordset
    Dim qySteps As DAO.QueryDef
    Dim recWspParams As Recordset
    Dim qyParams As DAO.QueryDef
    Dim recWspConns As Recordset
    Dim qyConns As DAO.QueryDef
    Dim recWspConnDtls As Recordset
    Dim qyConnDtls As DAO.QueryDef

    On Error GoTo LoadRunDataErr

    Set mcRunSteps.StepDB = mdbsLoadDb
    Set mcRunParams.ParamDatabase = mdbsLoadDb
    Set mcRunConstraints.ConstraintDB = mdbsLoadDb
    Set mcRunConnections.ConnDb = mdbsLoadDb
    Set mcRunConnDtls.ConnDb = mdbsLoadDb

    ' Read all the step and parameter data for the workspace
    Call ReadWorkspaceData(mlngWorkspaceId, mcRunSteps, _
            mcRunParams, mcRunConstraints, mcRunConnections,
mcRunConnDtls, _
            recWspSteps, qySteps, recWspParams, qyParams, recWspConns,
qyConns, _
            recWspConnDtls, qyConnDtls)

    ' Load all the pre- and post-execution constraints that
    ' have been defined for the workspace
    mcvntWspPreCons = mcRunConstraints.ConstraintsForWsp( _
        mlngWorkspaceId, _
        gintPreStep, _
        blnSort:=True, _
        blnGlobalConstraintsOnly:=True)
    mcvntWspPostCons = mcRunConstraints.ConstraintsForWsp( _
        mlngWorkspaceId, _
        gintPostStep, _
        blnSort:=True, _
        blnGlobalConstraintsOnly:=True)

    On Error Resume Next
    recWspSteps.Close
    qySteps.Close
    recWspParams.Close
    qyParams.Close
    recWspConns.Close
    qyConns.Close

    LoadRunData = True

    Exit Function

LoadRunDataErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ShowError errLoadRunDataFailed
    LoadRunData = False

End Function
Public Sub StopRun()

    On Error GoTo StopRunErr

    If mcRun Is Nothing Then
        ' We haven't been the run yet, so do nothing
    Else
        mcRun.StopRun
    End If

    Exit Sub

StopRunErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' Errors would have been displayed by the called process

End Sub
Public Sub AbortRun()

    On Error GoTo AbortRunErr

    If mcRun Is Nothing Then
        ' We haven't been the run yet, so do nothing
    Else
        mcRun.Abort
    End If

    Exit Sub

AbortRunErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' Errors would have been displayed by the called process

End Sub

Private Sub Class_Initialize()
```

```
    ' Create instances of the step, parameter and constraint arrays
    Set mcRunSteps = New cArrSteps
    Set mcRunParams = New cArrParameters
    Set mcRunConstraints = New cArrConstraints
    Set mcRunConnections = New cConnections
    Set mcRunConnDtls = New cConnDtls
    Set mcRun = New cRunInst
    Set mField = New cStringSM

End Sub
Private Sub Class_Terminate()

    On Error GoTo UnLoadRunDataErr

    ' Clears the step, parameter and constraint arrays
    Set mcRunSteps = Nothing
    Set mcRunParams = Nothing
    Set mcRunConstraints = Nothing
    Set mcRunConnections = Nothing
    Set mcRunConnDtls = Nothing

    Set mcRun = Nothing
    Set mdbsLoadDb = Nothing
    Set mField = Nothing

    Exit Sub

UnLoadRunDataErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' Not a critical error - continue
    Resume Next

End Sub

Private Sub mcRun_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    RaiseEvent ProcessComplete(cStepRecord, dtmEndTime, lngInstanceId)
    Call ProcessCompleteProcessing(cStepRecord, dtmEndTime,
lngInstanceId, lElapsed)

End Sub

Private Sub mcRun_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As
Long, _
        lParentInstanceId As Long, sItValue As String)

    RaiseEvent ProcessStart(cStepRecord, strCommand, dtmStartTime,
lngInstanceId)
    Call ProcessStartProcessing(cStepRecord, strCommand, dtmStartTime,
lngInstanceId, _
            lParentInstanceId, sItValue)

End Sub

Private Sub mcRun_RunComplete(dtmEndTime As Currency)

    Debug.Print "Run ended at: " & CStr(dtmEndTime)
    Call RunCompleteProcessing(dtmEndTime)

    RaiseEvent RunComplete(dtmEndTime)

End Sub
Private Sub mcRun_RunStart(dtmStartTime As Currency, strWspLog As
String)

    RaiseEvent RunStart(dtmStartTime, strWspLog, mlngRunId)
    Debug.Print "Run started at: " & CStr(dtmStartTime)

    Call RunStartProcessing(dtmStartTime)

End Sub
Private Sub mcRun_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    RaiseEvent StepComplete(cStepRecord, dtmEndTime, lngInstanceId)
'    BugMessage "Step: " & cStepRecord.StepLabel & " has completed!"

    Call StepCompleteProcessing(cStepRecord, dtmEndTime, lngInstanceId,
lElapsed)

End Sub
Private Sub mcRun_StepStart(cStepRecord As cStep, dtmStartTime As
Currency, _
        lngInstanceId As Long, lParentInstanceId As Long, sPath As
String, sIts As String, sItValue As String)

    RaiseEvent StepStart(cStepRecord, dtmStartTime, lngInstanceId, sPath,
sIts)
    'bugmessage "Step: " & cStepRecord.StepLabel & " has started."

    Call StepStartProcessing(cStepRecord, dtmStartTime, lngInstanceId,
lParentInstanceId, sItValue)

End Sub
```

cSequence.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cSequence"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cSequence.cls
'               Microsoft TPC-H Kit Ver. 1.00
```

```
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This class uses the att_identifiers table to generate
unique
'               identifiers.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mlngIdentifier As Long
Private mstrIdentifierColumn As String
Private mrecIdentifiers As Recordset
Private mdbsDatabase As Database

Private Const mstrEmptyString = ""

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cSequence."

Private Sub CreateIdRecord()
    ' Creates a record with all identifiers having an initial value of 1

    Dim sSql As String
    Dim pId As DAO.Parameter
    Dim qyId As DAO.QueryDef

    sSql = "insert into att_identifiers (" & _
            " workspace_id, parameter_id, step_id, " & _
            " constraint_id, run_id, connection_id " & _
            ", " & FLD_ID_CONN_NAME & _
            " ) values ( " & _
            "[w_id], [p_id], [s_id], [c_id], [r_id], [conn_id],
[conn_dtl_id] )"
    Set qyId = mdbsDatabase.CreateQueryDef(gstrEmptyString, sSql)
    For Each pId In qyId.Parameters
        pId.Value = glMinId
    Next pId
    qyId.Execute dbFailOnError
    qyId.Close

End Sub
Private Sub CreateIdRecordset()

    Dim strSql As String

    ' Initialize the recordset with all identifiers
    strSql = "select * from att_identifiers"
    Set mrecIdentifiers = mdbsDatabase.OpenRecordset(strSql,
dbOpenForwardOnly)

    If mrecIdentifiers.RecordCount = 0 Then
        CreateIdRecord
        Set mrecIdentifiers = mdbsDatabase.OpenRecordset(strSql,
dbOpenForwardOnly)
    End If

    BugAssert mrecIdentifiers.RecordCount <> 0

End Sub

Public Property Set IdDatabase(vdata As Database)

    Set mdbsDatabase = vdata

End Property

Public Property Let IdentifierColumn(vdata As String)

    Dim intIndex As Integer

    On Error GoTo IdentifierColumnErr

    ' Initialize the return value to an empty string
    mstrIdentifierColumn = mstrEmptyString
    Call CreateIdRecordset

    For intIndex = 0 To mrecIdentifiers.Fields.Count - 1

        If LCase(Trim(mrecIdentifiers.Fields(intIndex).Name)) = _
                LCase(Trim(vdata)) Then

            ' Valid column name
            mstrIdentifierColumn = vdata
            Exit Property
        End If

    Next intIndex

    BugAssert True, "Invalid column name!"

    Exit Property

IdentifierColumnErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "IdentifierColumn"
    On Error GoTo 0
    Err.Raise vbObjectError + errIdentifierColumnFailed, _
            mstrSource, _
            LoadResString(errIdentifierColumnFailed)

End Property
Public Property Get Identifier() As Long
    Dim strSql As String

    On Error GoTo GetIdentifierErr

    BugAssert mstrIdentifierColumn <> mstrEmptyString
```

```
    ' Increment the identifier column by 1
    strSql = "update att_identifiers " & _
        " set " & mstrIdentifierColumn & _
        " = " & mstrIdentifierColumn & " + 1"
    mdbsDatabase.Execute strSql, dbFailOnError

    ' Refresh the recordset with identifier values
    Call CreateIdRecordset

    mlngIdentifier = mrecIdentifiers.Fields(mstrIdentifierColumn).Value

    Identifier = mlngIdentifier

    Exit Property

GetIdentifierErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Identifier"
    On Error GoTo 0
    Err.Raise vbObjectError + errGetIdentifierFailed, _
            mstrSource, _
            LoadResString(errGetIdentifierFailed)

End Property
Private Sub Class_Terminate()

    mrecIdentifiers.Close

End Sub
```

cStack.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cStack"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cStack.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This class implements a stack of objects.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cStack."
Private mstrSource As String

Private mcVector As cVector
Private mlngCount As Long
Public Property Get Item(ByVal Position As Long) As Object
Attribute Item.VB_UserMemId = 0

    Set Item = mcVector(Position)

End Property

Public Sub Push(objToPush As Object)

    mcVector.Add objToPush

End Sub
Public Sub Clear()

    mcVector.Clear

End Sub

Public Function Pop() As Object

    If mcVector.Count > 0 Then
        Set Pop = mcVector.Delete(mcVector.Count - 1)
    Else
        Set Pop = Nothing
    End If

End Function
Public Function Count() As Long

    Count = mcVector.Count

End Function


Private Sub Class_Initialize()

    Set mcVector = New cVector

End Sub


Private Sub Class_Terminate()

    Set mcVector = Nothing

End Sub
```

cStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
```

```
END
Attribute VB_Name = "cStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Attribute VB_Ext_KEY = "SavedWithClassBuilder" ,"Yes"
Attribute VB_Ext_KEY = "Top_Level" ,"Yes"
'  FILE:        cStep.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a step.
'               Contains functions to insert, update and delete
'               att_steps records from the database.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Local variable(s) to hold property value(s)
Private mlngStepId As Long
Private mstrVersionNo As String
Private mstrStepLabel As String
Private mstrStepTextFile As String
Private mstrStepText As String
Private mstrStartDir As String
Private mlngWorkspaceId As Integer
Private mlngParentStepId As Integer
Private mstrParentVersionNo As String
Private mintSequenceNo As Integer
Private mintStepLevel As Integer
Private mblnEnabledFlag As Boolean
Private mstrDegreeParallelism As String
Private mintExecutionMechanism As Integer
Private mstrFailureDetails As String
Private mintContinuationCriteria As Integer
Private mblnGlobalFlag As Boolean
Private mblnArchivedFlag As Boolean
Private mstrOutputFile As String
'Private mstrLogFile As String
Private mstrErrorFile As String
Private mdbsDatabase As Database
Private mintStepType As Integer
Private mintOperation As Operation
Private mlngPosition As Long
Private mstrIteratorName As String
Private mcIterators As cNodeCollections
Private mbIsNewVersion As Boolean
Private msOldVersion As String

' The following constants are used throughout the project to
' indicate the different options selected by the user
' The options are presented to the user as control arrays of
' option buttons. These constants have to be in sync with the
' indexes of the option buttons.
' All the control arrays have an lbound of 1. The value 0 is
' used to indicate that the property being represented by the
' control array is not valid for the step
' Public enums are used since we cannot expose public constants
' in class modules. gintNoOption is applicable to all enums,
' but declared in the Execution method enum, since we cannot
' declare it more than once.

' Is here as a comment
' Has been defined in public.bas with the other object types
'Public Enum gintStepType
'    gintGlobalStep = 3
'    gintManagerStep
'    gintWorkerStep
'End Enum

' Execution Method options
Public Enum ExecutionMethod
    gintNoOption = 0
    gintExecuteODBC
    gintExecuteShell
End Enum

' Failure criteria options
Public Enum FailureCriteria
    gintFailureODBC = 1
    gintFailureTextCompare
End Enum

' Continuation criteria options
' Note: Update the initialization of gsContCriteria in Initialize() if
the
' continuation criteria are modified
Public Enum ContinuationCriteria
    gintOnFailureAbort = 1
    gintOnFailureContinue
    gintOnFailureCompleteSiblings
    gintOnFailureAbortSiblings
    gintOnFailureSkipSiblings
    gintOnFailureAsk
End Enum

' The initial version #
Private Const mstrMinVersion As String = "0.0"

' End of constants for option button control arrays
' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cStep."

' The cSequence class is used to generate unique step identifiers
Private mStepSeq As cSequence

' The StringSM class is used to carry out string operations
```

```
Private mFieldValue As cStringSM
Private Sub NewVersion()

mbIsNewVersion = True
msOldVersion = mstrVersionNo

End Sub
Public Function IsNewVersion() As Boolean
    IsNewVersion = mbIsNewVersion
End Function

Public Function OldVersionNo() As String
    OldVersionNo = msOldVersion
End Function

Public Sub SaveIterators()
    ' This procedure checks if any changes have been made
    ' to the iterators for the step. If so, it calls the
    ' methods of the iterator class to commit the changes
    Dim cItRec As cIterator
    Dim lngIndex As Long

    On Error GoTo SaveIteratorsErr

    For lngIndex = 0 To mcIterators.Count - 1
        Set cItRec = mcIterators(lngIndex)

        Select Case cItRec.IndOperation
            Case QueryOp
                ' No changes were made to the queried Step.
                ' Do nothing

            Case InsertOp
                cItRec.Add mlngStepId, mstrVersionNo
                cItRec.IndOperation = QueryOp

            Case UpdateOp
                cItRec.Update mlngStepId, mstrVersionNo
                cItRec.IndOperation = QueryOp

            Case DeleteOp
                cItRec.Delete mlngStepId, mstrVersionNo
                ' Remove the record from the collection
                mcIterators.Delete lngIndex

        End Select
    Next lngIndex

    Exit Sub

SaveIteratorsErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "SaveIterators"
    On Error GoTo 0
    Err.Raise vbObjectError + errSaveFailed, _
            mstrSource, _
            LoadResString(errSaveFailed)

End Sub
Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property
Public Property Let IndOperation(ByVal vdata As Operation)

    BugAssert vdata = QueryOp Or vdata = InsertOp Or vdata = UpdateOp Or
vdata = DeleteOp, "Invalid operation"
    mintOperation = vdata

End Property

Public Function Iterators() As Variant
    ' Returns a variant containing all the iterators that
    ' have been defined for the step

    Dim cStepIterators() As cIterator
    Dim cTempIt As cIterator
    Dim lngIndex As Long
    Dim lngItCount As Long

    On Error GoTo IteratorsErr

    lngItCount = 0
    For lngIndex = 0 To mcIterators.Count - 1
        ' Increase the array dimension and add the constraint
        ' to it
        Set cTempIt = mcIterators(lngIndex)

        If cTempIt.IndOperation <> DeleteOp Then
            ReDim Preserve cStepIterators(lngItCount)
            Set cStepIterators(lngItCount) = cTempIt
            lngItCount = lngItCount + 1
        End If

    Next lngIndex

    If lngItCount = 0 Then
        Iterators = Empty
    Else
        Iterators = cStepIterators()
    End If

    Call QuickSort(Iterators)

    Exit Function

IteratorsErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errIteratorsFailed, _
            mstrModuleName & "Iterators", _
```

```
            LoadResString(errIteratorsFailed)

End Function
Public Function IteratorCount() As Long
    ' Returns a count of all the iterators for the step

    Dim lngItCount As Long
    Dim lngIndex As Long
    Dim cTempIt As cIterator

    On Error GoTo IteratorsErr

    lngItCount = 0
    For lngIndex = 0 To mcIterators.Count - 1

        If mcIterators(lngIndex).IndOperation <> DeleteOp Then
            lngItCount = lngItCount + 1
        End If

    Next lngIndex

    IteratorCount = lngItCount

    Exit Function

IteratorsErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errIteratorsFailed, _
            mstrSource, _
            LoadResString(errIteratorsFailed)

End Function
Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    ' Check if the step label has been specified
    If StringEmpty(mstrStepLabel) Then
        ShowError errStepLabelMandatory
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                "Validate", LoadResString(errValidateFailed)
    End If

    If Not IsStringEmpty(mstrStepText) And Not
IsStringEmpty(mstrStepTextFile) Then
        ShowError errStepTextOrFile
        On Error GoTo 0
        Err.Raise vbObjectError + errStepTextOrFile, _
            "Validate", LoadResString(errStepTextOrFile)
    End If

End Sub

Public Function IncVersionY() As String
    ' The version number for a step is stored in the x.y
    ' format where x is the parent component and y is the
    ' child component of the step. This function will increment
    ' the y component of the step by 1

    On Error GoTo IncVersionYErr

    ' Store the old version number for the step
    Call NewVersion

    mstrVersionNo = Trim$(Str$(GetX(mstrVersionNo))) & gstrVerSeparator
& _
            Trim$(Str(GetY(mstrVersionNo) + 1))
    IncVersionY = mstrVersionNo

    Exit Function

IncVersionYErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "IncVersionY"
    On Error GoTo 0
    Err.Raise vbObjectError + errIncVersionYFailed, _
            gstrSource, _
            LoadResString(errIncVersionYFailed)

End Function
Public Function IncVersionX() As String
    ' The version number for a step is stored in the x.y
    ' format where x is the parent component and y is the
    ' child component of the step. This function will increment
    ' the y component of the step by 1 and reset the x component
    ' to 0

    On Error GoTo IncVersionXErr

    ' Store the old version number for the step
    Call NewVersion

    mstrVersionNo = Trim$(Str$(GetX(mstrVersionNo) + 1)) &
gstrVerSeparator & "0"
    IncVersionX = mstrVersionNo

    Exit Function

IncVersionXErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "IncVersionX"
    On Error GoTo 0
    Err.Raise vbObjectError + errIncVersionXFailed, _
            gstrSource, _
            LoadResString(errIncVersionXFailed)
```

```
End Function


Private Function GetY(strVersion As String) As Long
    ' The version number for a step is stored in the x.y
    ' format where x is the parent component and y is the
    ' child component of the step. Given an argument of type
    ' x.y, it returns y

    ' Truncate the fractional part to get the parent component
    ' of the version number (x.y)
    GetY = Val(Mid(strVersion, InStr(strVersion, gstrVerSeparator) + 1))

End Function
Private Function GetX(strVersion As String) As Long
    ' The version number for a step is stored in the x.y
    ' format where x is the parent component and y is the
    ' child component of the step. Given an argument of type
    ' x.y, it returns x

    ' Truncate the fractional part to get the parent component
    ' of the version number (x.y)
    GetX = Val(Left(strVersion, InStr(strVersion, gstrVerSeparator) -
1))

End Function

Public Function Clone(Optional cCloneStep As cStep) As cStep

    ' Creates a copy of a given step

    Dim lngIndex As Long
    Dim cItRec As cIterator
    Dim cItClone As cIterator

    On Error GoTo CloneErr

    If cCloneStep Is Nothing Then
        Set cCloneStep = New cStep
    End If

    ' Copy all the step properties to the newly created step
    ' Initialize the global flag first since subsequent
    ' validations might depend on it
    cCloneStep.GlobalFlag = mblnGlobalFlag
'    cCloneStep.GlobalRunMethod = mintGlobalRunMethod

    cCloneStep.StepType = mintStepType
    cCloneStep.StepId = mlngStepId
    cCloneStep.VersionNo = mstrVersionNo
    cCloneStep.StepLabel = mstrStepLabel
    cCloneStep.StepTextFile = mstrStepTextFile
    cCloneStep.StepText = mstrStepText
    cCloneStep.StartDir = mstrStartDir
    cCloneStep.WorkspaceId = mlngWorkspaceId
    cCloneStep.ParentStepId = mlngParentStepId
    cCloneStep.ParentVersionNo = mstrParentVersionNo
    cCloneStep.StepLevel = mintStepLevel
    cCloneStep.SequenceNo = mintSequenceNo
    cCloneStep.EnabledFlag = mblnEnabledFlag
    cCloneStep.DegreeParallelism = mstrDegreeParallelism
    cCloneStep.ExecutionMechanism = mintExecutionMechanism
    cCloneStep.FailureDetails = mstrFailureDetails
    cCloneStep.ContinuationCriteria = mintContinuationCriteria
    cCloneStep.ArchivedFlag = mblnArchivedFlag
    cCloneStep.OutputFile = mstrOutputFile
'    cCloneStep.LogFile = mstrLogFile
    cCloneStep.ErrorFile = mstrErrorFile
    cCloneStep.IteratorName = mstrIteratorName

    cCloneStep.IndOperation = mintOperation
    cCloneStep.Position = mlngPosition

    Set cCloneStep.NodeDB = mdbsDatabase

    ' Clone all the iterators for the step
    For lngIndex = 0 To mcIterators.Count - 1
        Set cItRec = mcIterators(lngIndex)
        Set cItClone = cItRec.Clone
        cCloneStep.LoadIterator cItClone
    Next lngIndex

    ' And set the return value to the newly created step
    Set Clone = cCloneStep

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, _
            mstrSource, LoadResString(errCloneFailed)

End Function
'End Sub
'
Public Property Let OutputFile(ByVal vdata As String)

    mstrOutputFile = vdata

End Property

Public Property Get OutputFile() As String

    OutputFile = mstrOutputFile

End Property

'Public Property Let LogFile(ByVal vdata As String)
'
```

```
'    mstrLogFile = vdata
'
'End Property
'
'Public Property Get LogFile() As String
'
'    LogFile = mstrLogFile
'
'End Property

Public Property Let ErrorFile(ByVal vdata As String)

    mstrErrorFile = vdata

End Property
Public Property Let IteratorName(ByVal vdata As String)

    mstrIteratorName = vdata

End Property

Public Property Get ErrorFile() As String

    ErrorFile = mstrErrorFile

End Property
Public Property Get IteratorName() As String

    IteratorName = mstrIteratorName

End Property

Public Property Set NodeDB(vdata As Database)

    Set mdbsDatabase = vdata
    Set mcIterators.NodeDB = vdata

End Property
Public Property Get NodeDB() As Database

    Set NodeDB = mdbsDatabase

End Property

Private Function IsStringEmpty(strToCheck As String) As Boolean

    IsStringEmpty = (strToCheck = gstrEmptyString)

End Function

Public Property Let EnabledFlag(ByVal vdata As Boolean)

    ' The enabled flag must be False for all global steps.
    ' This check must be made by the global step class. Only
    ' generic step validations will be carried out by this
    ' class
    mblnEnabledFlag = vdata

End Property

Public Property Let GlobalFlag(ByVal vdata As Boolean)

    mblnGlobalFlag = vdata

End Property
Public Property Get EnabledFlag() As Boolean

    EnabledFlag = mblnEnabledFlag

End Property

Public Property Let ArchivedFlag(ByVal vdata As Boolean)

    mblnArchivedFlag = vdata

End Property

Public Property Get ArchivedFlag() As Boolean

    ArchivedFlag = mblnArchivedFlag

End Property

Public Property Get GlobalFlag() As Boolean

    GlobalFlag = mblnGlobalFlag

End Property

Public Sub Add()
    ' Inserts a step record into the database - it initializes
    ' the necessary properties for the step and calls InsertStepRec
    ' to do the database work

    On Error GoTo AddErr

    ' A new record would have the deleted_flag turned off!
    mblnArchivedFlag = False

    Call InsertStepRec

    ' If a new version of a step has been created, reset the old version
info, since
    ' it's already been saved to the db
    If IsNewVersion() Then
        mbIsNewVersion = False
        msOldVersion = gstrEmptyString
    End If

    Exit Sub

AddErr:
```

```
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errAddStepFailed, _
            mstrModuleName & "Add", LoadResString(errAddStepFailed)
End Sub
Private Sub InsertStepRec()
    ' Inserts a step record into the database
    ' It first generates the insert statement using the different
    ' step properties and then executes it

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo InsertStepRecErr

    ' First check if the database object is valid
    Call CheckDB

    ' Check if the step record is valid
    Call Validate

    If IsNewVersion() Then
        Call UpdOldVersionsArchFlg
    End If

    ' Create a temporary querydef object
    strInsert = "insert into att_steps " & _
        "( workspace_id, step_id, version_no, " & _
        " step_label, step_file_name, step_text, start_directory, " & _
        " parent_step_id, parent_version_no, sequence_no, " & _
        " enabled_flag, step_level, " & _
        " degree_parallelism, execution_mechanism, " & _
        " failure_details, " & _
        " continuation_criteria, global_flag, " & _
        " archived_flag, " & _
        " output_file_name, error_file_name, " & _
        " iterator_name ) values ( "

    ' log_file_name,

#If USE_JET Then

    strInsert = strInsert & " [w_id], [s_id], [ver_no], " & _
        " [s_label], [s_file_name], [s_text], [s_start_dir], " & _
        " [p_step_id], [p_version_no], [seq_no], " & _
        " [enabled], [s_level], [deg_parallelism], " & _
        " [exec_mechanism], [fail_dtls], " & _
        " [cont_criteria], [global], [archived], " & _
        " [output_file], [error_file], " & _
        " [it_name] ) "

    ' [log_file],

    Set qy = mdbsDatabase.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close
#Else

        strInsert = strInsert & Str(mlngWorkspaceId) & ", " &
Str(mlngStepId) & _
        ", " & mFieldValue.MakeStringFieldValid(mstrVersionNo)

    ' For fields that may be null, call a function to determine
    ' the string to be appended to the insert statement
    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrStepLabel)
    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrStepTextFile)
    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrStepText)
    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrStartDir)

    strInsert = strInsert & ", " & Str(mlngParentStepId) & _
        ", " & mFieldValue.MakeStringFieldValid(mstrParentVersionNo) & _
        ", " & Str(mintSequenceNo) & _
        ", " & Str(mblnEnabledFlag) & ", " & Str(mintStepLevel)

    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrDegreeParallelism)
    strInsert = strInsert & ", " & Str(mintExecutionMechanism)

    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrFailureDetails) & _
        ", " & Str(mintContinuationCriteria) & _
        ", " & Str(mblnGlobalFlag) & _
        ", " & Str(mblnArchivedFlag)

    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrOutputFile)
'    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrLogFile)
    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrErrorFile)
    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrIteratorName)

    strInsert = strInsert & " ) "

    BugMessage strInsert
    mdbsDatabase.Execute strInsert, dbFailOnError

#End If

    Exit Sub

InsertStepRecErr:
    mstrSource = mstrModuleName & "InsertStepRec"
```

```
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertStepFailed, _
            mstrSource, LoadResString(errInsertStepFailed)
End Sub
Private Sub UpdOldVersionsArchFlg()
    ' Updates the archived flag on all old version for the step to True

    Dim sUpdate As String
    Dim qy As DAO.QueryDef

    On Error GoTo UpdOldVersionsArchFlgErr
    mstrSource = mstrModuleName & "UpdOldVersionsArchFlg"

#If USE_JET Then

    sUpdate = "update att_steps " & _
        " set archived_flag = True "

    ' Append the Where clause
    sUpdate = sUpdate & " where step_id = [s_id] " & _
        " and version_no <> [ver_no]"

    Set qy = mdbsDatabase.CreateQueryDef(gstrEmptyString, sUpdate)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    If qy.RecordsAffected = 0 Then
        On Error GoTo 0
        Err.Raise vbObjectError + errModifyStepFailed, _
                mstrSource, LoadResString(errModifyStepFailed)
    End If

    qy.Close

#Else

    sUpdate = "update att_steps " & _
        " set archived_flag = True "

    sUpdate = sUpdate & " where step_id = " & Str(mlngStepId) & _
        " and version_no <> " &
mFieldValue.MakeStringFieldValid(mstrVersionNo)

    BugMessage sUpdate
    mdbsDatabase.Execute sUpdate, dbFailOnError
#End If

    Exit Sub

UpdOldVersionsArchFlgErr:
    mstrSource = mstrModuleName & "UpdOldVersionsArchFlg"
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyStepFailed, _
            mstrSource, LoadResString(errModifyStepFailed)
End Sub
Public Sub InsertIterator(cItRecord As cIterator)
    ' Inserts the iterator record into the database

    Call cItRecord.Add(mlngStepId, mstrVersionNo)

End Sub
Public Sub UpdateIterator(cItRecord As cIterator)
    ' Updates the iterator record in the database

    Call cItRecord.Update(mlngStepId, mstrVersionNo)

End Sub
Public Sub UpdateIteratorVersion()
    ' Updates the iterator record in the database

    Dim lngIndex As Long
    Dim cTempIt As cIterator

    On Error GoTo UpdateIteratorVersionErr

    For lngIndex = 0 To mcIterators.Count - 1
        ' Increase the array dimension and add the constraint
        ' to it
        Set cTempIt = mcIterators(lngIndex)

        If cTempIt.IndOperation <> DeleteOp Then
            ' Set the operation to indicate an insert
            cTempIt.IndOperation = InsertOp
        End If

    Next lngIndex

    Exit Sub

UpdateIteratorVersionErr:
    mstrSource = mstrModuleName & "UpdateIteratorVersion"
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateFailed, _
            mstrSource, LoadResString(errUpdateFailed)

End Sub
Public Sub AddIterator(cItRecord As cIterator)
    ' Adds the iterator record to the collection of iterators
    ' for the step

    Call mcIterators.Add(cItRecord)

End Sub
Public Sub AddAllIterators()
    ' Sets the indicator variable for all iterators to insert

    Dim lngIndex As Long
```

```
    For lngIndex = 0 To mcIterators.Count - 1
        mcIterators(lngIndex).Validate
        mcIterators(lngIndex).IndOperation = InsertOp
    Next lngIndex

End Sub

Public Sub LoadIterator(cItRecord As cIterator)
    ' Adds the iterator record to the collection of iterators
    ' for the step

    Call mcIterators.Load(cItRecord)

End Sub
Public Sub UnloadIterators()
    ' Unloads all iterator records for the step

    Dim lngIndex As Long

    For lngIndex = mcIterators.Count - 1 To 0 Step -1
        ' Calls the collection method to unload the node
        ' from the array
        mcIterators.Unload lngIndex
    Next lngIndex

End Sub
Public Sub ModifyIterator(cItRecord As cIterator)
    ' Modifies the iterator record in the collection

    Call mcIterators.Modify(cItRecord)

End Sub
Public Sub DeleteIterator(cItRecord As cIterator)
    ' Deletes the iterator record from the database

    Call cItRecord.Delete(mlngStepId, mstrVersionNo)

End Sub
Public Sub RemoveIterator(cItRecord As cIterator)
    ' Marks the iterator record in the collection to
    ' indicate a delete

    Call mcIterators.Delete(cItRecord.Position)

End Sub


Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the actual field names. When the parameter names
    ' are the same as the field names, parameters in the
    ' where clause do not get created.

    Dim prmParam As DAO.Parameter

On Error GoTo AssignParametersErr
mstrSource = mstrModuleName & "AssignParameters"

For Each prmParam In qyExec.Parameters
    Select Case prmParam.Name
        Case "[w_id]"
            prmParam.Value = mlngWorkspaceId
        Case "[s_id]"
            prmParam.Value = mlngStepId
        Case "[ver_no]"
            prmParam.Value = mstrVersionNo
        Case "[s_label]"
            prmParam.Value = mstrStepLabel
        Case "[s_file_name]"
            prmParam.Value = mstrStepTextFile
        Case "[s_text]"
            prmParam.Value = mstrStepText
        Case "[s_start_dir]"
            prmParam.Value = mstrStartDir
        Case "[p_step_id]"
            prmParam.Value = mlngParentStepId
        Case "[p_version_no]"
            prmParam.Value = mstrParentVersionNo
        Case "[seq_no]"
            prmParam.Value = mintSequenceNo
        Case "[enabled]"
            prmParam.Value = mblnEnabledFlag
        Case "[s_level]"
            prmParam.Value = mintStepLevel
        Case "[deg_parallelism]"
            prmParam.Value = mstrDegreeParallelism
        Case "[exec_mechanism]"
            prmParam.Value = mintExecutionMechanism
        Case "[fail_dtls]"
            prmParam.Value = mstrFailureDetails
        Case "[cont_criteria]"
            prmParam.Value = mintContinuationCriteria
        Case "[global]"
            prmParam.Value = mblnGlobalFlag
        Case "[archived]"
            prmParam.Value = mblnArchivedFlag
        Case "[output_file]"
            prmParam.Value = mstrOutputFile
'           Case "[log_file]"
'               prmParam.Value = mstrLogFile
        Case "[error_file]"
            prmParam.Value = mstrErrorFile
        Case "[it_name]"
            prmParam.Value = mstrIteratorName
        Case Else
            ' Write the parameter name that is faulty
            WriteError errInvalidParameter, mstrSource, _
                    prmParam.Name
            On Error GoTo 0
            Err.Raise errInvalidParameter, mstrSource, _
```

```
                        LoadResString(errInvalidParameter)
        End Select
    Next prmParam

    If qyExec.Parameters("s_id") = 0 Or
StringEmpty(qyExec.Parameters("ver_no")) Then
        WriteError errInvalidParameter, mstrSource
        On Error GoTo 0
        Err.Raise errInvalidParameter, mstrSource,
LoadResString(errInvalidParameter)
    End If

    Exit Sub

AssignParametersErr:

    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrSource, LoadResString(errAssignParametersFailed)

End Sub

Public Sub Modify()

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo ModifyErr
    mstrSource = mstrModuleName & "Modify"

    ' Check if the database object is valid
    Call CheckDB

    ' Check if the step record is valid
    Call Validate

    ' The step_id and version_no will never be updated -
    ' whenever a step is modified a copy of the old step will
    ' be created with an incremented version_no

#If USE_JET Then

    strUpdate = "update att_steps " & _
        " set step_label = [s_label] " & _
        " , step_file_name = [s_file_name] " & _
        " , step_text = [s_text] " & _
        " , start_directory = [s_start_dir] " & _
        " , workspace_id = [w_id] " & _
        " , parent_step_id = [p_step_id] " & _
        " , parent_version_no = [p_version_no] " & _
        " , sequence_no = [seq_no] " & _
        " , step_level = [s_level] " & _
        " , enabled_flag = [enabled] " & _
        " , degree_parallelism = [deg_parallelism] " & _
        " , execution_mechanism = [exec_mechanism] " & _
        " , failure_details = [fail_dtls] " & _
        " , continuation_criteria = [cont_criteria] " & _
        " , global_flag = [global] " & _
        " , archived_flag = [archived] " & _
        " , output_file_name = [output_file] " & _
        " , error_file_name = [error_file] " & _
        " , iterator_name = [it_name] "

        ' " , log_file_name = [log_file] " & _

    ' Append the Where clause
    strUpdate = strUpdate & " where step_id = [s_id] " & _
        " and version_no = [ver_no]"

    Set qy = mdbsDatabase.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    If qy.RecordsAffected = 0 Then
        On Error GoTo 0
        Err.Raise vbObjectError + errModifyStepFailed, _
                mstrSource, LoadResString(errModifyStepFailed)
    End If

    qy.Close

#Else

    strUpdate = "update att_steps " & _
        " set step_label = "

    ' For fields that may be null, call a function to determine
    ' the string to be appended to the update statement
    strUpdate = strUpdate &
mFieldValue.MakeStringFieldValid(mstrStepLabel)

    strUpdate = strUpdate & " , step_file_name = " &
mFieldValue.MakeStringFieldValid(mstrStepTextFile)
    strUpdate = strUpdate & " , step_text = " &
mFieldValue.MakeStringFieldValid(mstrStepText)
    strUpdate = strUpdate & " , start_directory = " &
mFieldValue.MakeStringFieldValid(mstrStartDir)

    strUpdate = strUpdate & " , workspace_id = " & Str(mlngWorkspaceId)
& _
        " , parent_step_id = " & Str(mlngParentStepId) & _
        " , parent_version_no = " &
mFieldValue.MakeStringFieldValid(mstrParentVersionNo) & _
        " , sequence_no = " & Str(mintSequenceNo) & _
        " , step_level = " & Str(mintStepLevel) & _
        " , enabled_flag = " & Str(mblnEnabledFlag) & _
        " , degree_parallelism = " &
mFieldValue.MakeStringFieldValid(mstrDegreeParallelism) & _
```

```
        " , execution_mechanism = " & Str(mintExecutionMechanism) & _
        " , failure_details = " &
mFieldValue.MakeStringFieldValid(mstrFailureDetails) & _
        " , continuation_criteria = " & Str(mintContinuationCriteria) &
_
        " , global_flag = " & Str(mblnGlobalFlag) & _
        " , archived_flag = " & Str(mblnArchivedFlag) & _
        " , output_file_name = " &
mFieldValue.MakeStringFieldValid(mstrOutputFile) & _
        " , error_file_name = " &
mFieldValue.MakeStringFieldValid(mstrErrorFile) & _
        " , iterator_name = " &
mFieldValue.MakeStringFieldValid(mstrIteratorName)

'        " , log_file_name = " &
mFieldValue.MakeStringFieldValid(mstrLogFile) & _

    strUpdate = strUpdate & " where step_id = " & Str(mlngStepId) & _
        " and version_no = " &
mFieldValue.MakeStringFieldValid(mstrVersionNo)

    BugMessage strUpdate
    mdbsDatabase.Execute strUpdate, dbFailOnError
#End If

    Exit Sub

ModifyErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Modify"
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyStepFailed, _
            mstrSource, LoadResString(errModifyStepFailed)
End Sub
Private Sub CheckDB()
    ' Check if the database object has been initialized

    If mdbsDatabase Is Nothing Then
        ShowError errInvalidDB
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidDB, _
            mstrModuleName, LoadResString(errInvalidDB)
    End If

End Sub

Public Sub Delete()

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr

    Call CheckDB

    strDelete = "delete from att_steps " & _
            " where step_id = [s_id] " & _
            " and version_no = [ver_no] "
'    mdbsDatabase.Execute strDelete, dbFailOnError
    Set qy = mdbsDatabase.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteStepFailed, _
            mstrModuleName & "Delete",
LoadResString(errDeleteStepFailed)
End Sub
Public Property Get DegreeParallelism() As String

    DegreeParallelism = mstrDegreeParallelism

End Property
Public Property Get Position() As Long

    Position = mlngPosition

End Property

Public Property Let DegreeParallelism(ByVal vdata As String)

    ' The degree of parallelism must be zero for all global steps
    ' This check must be made by the global step class. Only
    ' generic step validations will be carried out by this
    ' class
    mstrDegreeParallelism = vdata

End Property

Public Property Let ExecutionMechanism(ByVal vdata As ExecutionMethod)

    BugAssert vdata = gintExecuteODBC Or vdata = gintExecuteShell Or
vdata = gintNoOption, _
            "Execution mechanism invalid"
    mintExecutionMechanism = vdata

End Property

Public Property Let FailureDetails(ByVal vdata As String)

    mstrFailureDetails = vdata

End Property
Public Property Let SequenceNo(ByVal vdata As Integer)
    mintSequenceNo = vdata
```

```
End Property

Public Property Let Position(ByVal vdata As Long)
    mlngPosition = vdata
End Property

Public Property Let ParentStepId(ByVal vdata As Long)
    mlngParentStepId = vdata
End Property

Public Property Get SequenceNo() As Integer

    SequenceNo = mintSequenceNo

End Property

Public Property Get StepLevel() As Integer
    StepLevel = mintStepLevel
End Property

Public Property Get ParentVersionNo() As String
    ParentVersionNo = mstrParentVersionNo
End Property

Public Property Let ParentVersionNo(ByVal vdata As String)
    mstrParentVersionNo = vdata
End Property

Public Property Get ParentStepId() As Long
    ParentStepId = mlngParentStepId
End Property

Public Property Let WorkspaceId(ByVal vdata As Long)
    mlngWorkspaceId = vdata
End Property

Public Property Let VersionNo(ByVal vdata As String)
    ' The version number of a step is stored in the x.y format where
    ' x represents a change to the step as a result of modifications
    ' to any of the step properties
    ' y represents a change to the step as a result of modifications
    ' to the sub-steps associated with it. Hence the y-component
    ' of the version will be incremented when a sub-step is added,
    ' modified or deleted
    ' x will be referred to throughout this code as the parent
    ' component of the version and y will be referred to as the
    ' child component of the version
    ' The version information for a step is maintained by the
    ' calling function

    mstrVersionNo = vdata

End Property

Public Property Get StepType() As gintStepType

    On Error GoTo StepTypeErr

    If mintStepType = 0 Then
        ' The step type variable has not been initialized -
        If mblnGlobalFlag Then
            mintStepType = gintGlobalStep
        ElseIf IsStringEmpty(mstrStepText) And _
                IsStringEmpty(mstrStepTextFile) Then
            mintStepType = gintManagerStep
        Else
            mintStepType = gintWorkerStep
        End If
    End If

    StepType = mintStepType

    Exit Property

StepTypeErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "StepType"
    On Error GoTo 0
    Err.Raise vbObjectError + errGetStepTypeFailed, _
            mstrSource, _
            LoadResString(errGetStepTypeFailed)

End Property

Public Property Let StepType(vdata As gintStepType)

    On Error GoTo StepTypeErr

    Select Case vdata
        Case gintGlobalStep, gintManagerStep, gintWorkerStep
            mintStepType = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errStepTypeInvalid, _
                    mstrModuleName & "StepType",
LoadResString(errStepTypeInvalid)
    End Select
    Exit Property

StepTypeErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "StepType"
    On Error GoTo 0
    Err.Raise vbObjectError + errLetStepTypeFailed, _
            mstrSource, _
            LoadResString(errLetStepTypeFailed)

End Property

Public Property Get WorkspaceId() As Long
    WorkspaceId = mlngWorkspaceId
```

```
End Property

Public Property Get ContinuationCriteria() As ContinuationCriteria

    ContinuationCriteria = mintContinuationCriteria

End Property

Public Property Let ContinuationCriteria(ByVal vdata As
ContinuationCriteria)

    ' The Continuation criteria must be null for all global steps
    ' and non-null for all manager and worker steps
    ' These checks will have to be made by the corresponding
    ' classes - only generic step validations will be made
    ' by this class
    BugAssert vdata = gintOnFailureAbortSiblings Or vdata =
gintOnFailureCompleteSiblings _
                Or vdata = gintOnFailureSkipSiblings Or vdata =
gintOnFailureAbort _
                Or vdata = gintOnFailureContinue Or vdata =
gintOnFailureAsk _
                Or vdata = gintNoOption, _
                "Invalid continuation criteria"
    mintContinuationCriteria = vdata

End Property
Public Property Get ExecutionMechanism() As ExecutionMethod

    ExecutionMechanism = mintExecutionMechanism

End Property


Public Property Get FailureDetails() As String

    FailureDetails = mstrFailureDetails

End Property

Public Property Let StepText(ByVal vdata As String)
    ' Has to be null for manager steps
    ' The check will have to be made by the user interface or
    ' by the manager step class
    mstrStepText = vdata
End Property
Public Property Let StepLevel(ByVal vdata As Integer)

    ' The step level must be zero for all global steps
    ' This check must be made in the global step class
    mintStepLevel = vdata

End Property
Public Property Get StepText() As String
    StepText = mstrStepText
End Property

Public Property Let StepTextFile(ByVal vdata As String)
    ' Has to be null for manager steps
    ' The check will have to be made by the user interface and
    ' by the manager step class
    mstrStepTextFile = vdata
End Property

Public Property Get StepTextFile() As String
    StepTextFile = mstrStepTextFile
End Property

Public Property Let StepLabel(ByVal vdata As String)
    ' Cannot be null for manager steps
    ' But this check cannot be made here since we do not know
    ' at this point if the step being created is a manager
    ' or a worker step
    ' The check will have to be made by the user interface and
    ' by the manager step class
    mstrStepLabel = vdata
End Property

Public Property Get StepLabel() As String
    StepLabel = mstrStepLabel
End Property

Public Property Let StartDir(ByVal vdata As String)
    mstrStartDir = vdata
End Property

Public Property Get StartDir() As String
    StartDir = mstrStartDir
End Property

Public Property Get VersionNo() As String
    ' The version number of a step is stored in the x.y format where
    ' x represents a change to the step as a result of modifications
    ' to any of the step properties
    ' y represents a change to the step as a result of modifications
    ' to the sub-steps associated with it. Hence the y-component
    ' of the version will be incremented when a sub-step is added,
    ' modified or deleted
    ' x will be referred to throughout this code as the parent
    ' component of the version and y will be referred to as the
    ' child component of the version
    ' The version information for a step is maintained by the
    ' calling function

    VersionNo = mstrVersionNo

End Property

Public Property Get StepId() As Long

    StepId = mlngStepId
```

```
End Property
Public Property Get NextStepId() As Long

    Dim lngNextId As Long

    On Error GoTo NextStepIdErr

    ' First check if the database object is valid
    Call CheckDB

    ' Retrieve the next identifier using the sequence class
    Set mStepSeq = New cSequence
    Set mStepSeq.IdDatabase = mdbsDatabase
    mStepSeq.IdentifierColumn = "step_id"
    lngNextId = mStepSeq.Identifier
    Set mStepSeq = Nothing

    NextStepId = lngNextId
    Exit Property

NextStepIdErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "NextStepId"
    On Error GoTo 0
    Err.Raise vbObjectError + errStepIdGetFailed, _
        mstrSource, LoadResString(errStepIdGetFailed)

End Property
Public Property Let StepId(ByVal vdata As Long)

    mlngStepId = vdata

End Property

Private Sub Class_Initialize()

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp
    mbIsNewVersion = False
    msOldVersion = gstrEmptyString

    Set mFieldValue = New cStringSM
    Set mcIterators = New cNodeCollections

End Sub

Private Sub Class_Terminate()

    Set mFieldValue = Nothing
    Set mcIterators = Nothing

End Sub
```

cStepTree.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cStepTree"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cStepTree.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Implements step navigation functions such as determining
'               the child of a step and so on.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cStepTree."
Private mstrSource As String

Public StepRecords As cArrSteps
Public Property Get HasChild(Optional ByVal StepKey As String, _
        Optional ByVal StepId As Long = 0) As Boolean

    Dim lTemp As Long

    HasChild = False
    StepId = GetStepId(StepKey, StepId)

    For lTemp = 0 To StepRecords.StepCount - 1
        If StepRecords(lTemp).StepType <> gintGlobalStep And
StepRecords(lTemp).ParentStepId = StepId Then
            HasChild = True
            Exit For
        End If
    Next lTemp

End Property
Public Property Get ChildStep(Optional ByVal StepKey As String, _
        Optional ByVal StepId As Long = 0) As cStep

    Dim lTemp As Long

    Set ChildStep = Nothing
    StepId = GetStepId(StepKey, StepId)

    For lTemp = 0 To StepRecords.StepCount - 1
```

```
        If StepRecords(lTemp).StepType <> gintGlobalStep And
StepRecords(lTemp).ParentStepId = StepId And _
                StepRecords(lTemp).SequenceNo = gintMinSequenceNo Then
            Set ChildStep = StepRecords(lTemp)
            Exit For
        End If
    Next lTemp

End Property
Public Property Get NextStep(Optional ByVal StepKey As String, _
        Optional ByVal StepId As Long = 0) As cStep

    Dim lTemp As Long
    Dim cChildStep As cStep

    Set NextStep = Nothing
    StepId = GetStepId(StepKey, StepId)
    Set cChildStep = StepRecords.QueryStep(StepId)

    For lTemp = 0 To StepRecords.StepCount - 1
        If StepRecords(lTemp).StepType <> gintGlobalStep And _
                StepRecords(lTemp).ParentStepId =
cChildStep.ParentStepId And _
                StepRecords(lTemp).SequenceNo = cChildStep.SequenceNo +
1 Then
            Set NextStep = StepRecords(lTemp)
            Exit For
        End If
    Next lTemp

End Property
Private Function GetStepId(Optional ByVal StepKey As String, _
        Optional ByVal StepId As Long = 0) As Long
    If StepId = 0 Then
        If StringEmpty(StepKey) Then
            Err.Raise vbObjectError + errMandatoryParameterMissing, _
                    mstrModuleName & "GetStepId",
LoadResString(errMandatoryParameterMissing)
        Else
            GetStepId = IIf(IsLabel(StepKey), 0,
MakeIdentifierValid(StepKey))
        End If
    Else
        GetStepId = StepId
    End If
End Function
```

cStringSM.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cStringSM"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cStringSM.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module contains common procedures that can be used
'               to manipulate strings
'               It is called StringSM, since String is a Visual Basic
keyword
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cStringSM."

Private mstrText As String

Private Const mstrNullValue = "null"
Private Const mstrSQ = "'"
Private Const mstrEnvVarSeparator = "%"
Public Function InsertEnvVariables( _
        Optional ByVal strComString As String) As String
    ' This function replaces all environment variables in
    ' the passed in string with their values - they are
    ' enclosed by "%"

    Dim intPos As Integer
    Dim intEndPos As Integer
    Dim strEnvVariable As String
    Dim strValue As String
    Dim strCommand As String

    On Error GoTo InsertEnvVariablesErr
    mstrSource = mstrModuleName & "InsertEnvVariables"

    ' Initialize the return value of the function to the
    ' passed in command
    If IsStringEmpty(strComString) Then
        strCommand = mstrText
    Else
        strCommand = strComString
    End If

    intPos = InStr(strCommand, mstrEnvVarSeparator)
    Do While intPos <> 0
        ' Extract the environment variable from the passed
        ' in string
        intEndPos = InStr(intPos + 1, strCommand, mstrEnvVarSeparator)
        strEnvVariable = Mid(strCommand, intPos + 1, intEndPos - intPos
- 1)
```

```
        ' Get the value of the variable and call a function
        ' to replace the variable with it's value
        strValue = Environ$(strEnvVariable)
        strCommand = ReplaceSubString(strCommand, _
                mstrEnvVarSeparator & strEnvVariable &
mstrEnvVarSeparator, _
                strValue)

        intPos = InStr(strCommand, mstrEnvVarSeparator)
    Loop

    InsertEnvVariables = strCommand
    Exit Function

InsertEnvVariablesErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' Return an empty string
    InsertEnvVariables = gstrEmptyString

End Function
Public Function MakeStringFieldValid( _
        Optional strField As String = gstrEmptyString) As String
    ' Returns a string that can be appended to any insert
    ' or modify (sql) statement
    ' If an argument is not passed to this function, the
    ' default text property is used

    Dim strTemp As String

    On Error GoTo MakeStringFieldValidErr

    If IsStringEmpty(strField) Then
        strTemp = mstrText
    Else
        strTemp = strField
    End If

    ' It checks whether the text is empty
    ' If so, it returns the string, "null"
    If IsStringEmpty(strTemp) Then
        MakeStringFieldValid = mstrNullValue
    Else
        ' Single-quotes have to be replaced by two single-quotes,
        ' since a single-quote is the identifier delimiter
        ' character - call a procedure to do the replace
        strTemp = ReplaceSubString(strTemp, mstrSQ, mstrSQ & mstrSQ)

        ' Replace pipe characters with the corresponding chr function
        strTemp = ReplaceSubString(strTemp, "|", "' & Chr(124) & '")

        ' Enclose the string in single quotes
        MakeStringFieldValid = mstrSQ & strTemp & mstrSQ

    End If

    Exit Function

MakeStringFieldValidErr:
    mstrSource = mstrModuleName & "MakeStringFieldValid"
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errMakeFieldValidFailed, _
            mstrSource, LoadResString(errMakeFieldValidFailed)

End Function
Public Function MakeDateFieldValid( _
        Optional dtmField As Date = gdtmEmpty) As String
    ' Returns a string that can be appended to any insert
    ' or modify (sql) statement

    ' Enclose the date in single quotes
    MakeDateFieldValid = mstrSQ & dtmField & mstrSQ

End Function

Private Function IsStringEmpty(strToCheck As String) As Boolean

    If strToCheck = gstrEmptyString Then
        IsStringEmpty = True
    Else
        IsStringEmpty = False
    End If

End Function
Public Function ReplaceSubString(ByVal MainString As String, _
        ByVal ReplaceString As String, _
        ByVal ReplaceWith As String) As String

    ' Replaces all occurrences of ReplaceString in MainString with
ReplaceWith

    Dim intPos As Integer
    Dim strTemp As String

    On Error GoTo ReplaceSubStringErr

    strTemp = MainString

    intPos = InStr(strTemp, ReplaceString)
    Do While intPos <> 0
        strTemp = Left(strTemp, intPos - 1) & ReplaceWith & _
                Mid(strTemp, intPos + Len(ReplaceString))
        intPos = InStr(intPos + Len(ReplaceString) + 1, strTemp,
ReplaceString)
    Loop
    ReplaceSubString = strTemp

    Exit Function

ReplaceSubStringErr:
```

```
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "ReplaceSubString"
    On Error GoTo 0
    Err.Raise vbObjectError + errParseStringFailed, _
        mstrSource, _
        LoadResString(errParseStringFailed)

End Function

Public Property Get Text() As String
Attribute Text.VB_UserMemId = 0
    Text = mstrText
End Property

Public Property Let Text(ByVal vdata As String)
    mstrText = vdata
End Property
```

cSubStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cSubStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cSubStep.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module encapsulates the properties of sub-steps
'               that are used during the execution of a workspace.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cSubStep"

Private mlngStepId As Long
Private mintRunning As Integer ' Number of running tasks
Private mintComplete As Integer ' Number of completed tasks
' The last iterator for this sub-step
Private mcLastIterator As cRunItDetails

Public Function NewIteration(cStepRec As cStep) As cIterator
    ' Calls a procedure to determine the next iterator value
    ' for the passed in step - returns the value to be used
    ' in the iteration.
    ' It updates the instance node with the new iteration
    ' for the step.

    Dim cItRec As cIterator

    On Error GoTo NewIterationErr

    ' Call a function that will populate an iterator record
    ' with the iterator values
    Set cItRec = NextIteration(cStepRec)

    ' Initialize the run node with the new iterator
    ' values
    If Not mcLastIterator Is Nothing Then
        If cItRec Is Nothing Then
            mcLastIterator.Value = gstrEmptyString
        Else
            mcLastIterator.Value = cItRec.Value

            ' And if the iterator is a list of values, then update
            ' the sequence number as well
            If mcLastIterator.IteratorType = gintValue Then
                mcLastIterator.Sequence = cItRec.SequenceNo
            End If
        End If
    End If

    Set NewIteration = cItRec
    Set cItRec = Nothing

    Exit Function

NewIterationErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
        LoadResString(errIterateFailed)

End Function
Public Function NextIteration(cStepRec As cStep) As cIterator

    ' Retrieves the next iterator value for the passed in step -
    ' returns an iterator record with the new iterator values

    Dim cItRec As cIterator
    Dim vntIterators As Variant
    Dim lngValue As String

    On Error GoTo NextIterationErr

    vntIterators = cStepRec.Iterators

    If Not mcLastIterator Is Nothing Then
        ' The run node contains the iterator details
        ' Get the next value for the iterator
        If mcLastIterator.IteratorType = gintValue Then
```

```
            ' Find the next iterator that appears in the list of
            ' iterator values
            Set cItRec = NextInSequence(vntIterators,
mcLastIterator.Sequence)
        Else
            lngValue = CLng(Trim$(mcLastIterator.Value))
            ' Determine whether the new iterator value falls in the
            ' range between From and To
            If (mcLastIterator.RangeStep > 0 And _
                    (mcLastIterator.RangeFrom <= mcLastIterator.RangeTo)
And _
                    (mcLastIterator.RangeStep + lngValue) <=
mcLastIterator.RangeTo) Or _
                    (mcLastIterator.RangeStep < 0 And _
                    (mcLastIterator.RangeFrom >= mcLastIterator.RangeTo)
And _
                    (mcLastIterator.RangeStep + lngValue) >=
mcLastIterator.RangeTo) Then
                Set cItRec = New cIterator
                cItRec.Value = Trim$(CStr(mcLastIterator.RangeStep +
lngValue))
            Else
                Set cItRec = Nothing
            End If
        End If
    Else
        Set cItRec = Nothing
    End If

    Set NextIteration = cItRec
    Exit Function

NextIterationErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
        LoadResString(errIterateFailed)

End Function
Public Sub InitializeIt(cPendingStep As cStep, _
        ColParameters As cArrParameters, _
        Optional vntIterators As Variant)

    ' Initializes the LastIteration structure with the iterator details
for the
    ' passed in step

    On Error GoTo InitializeItErr

    If IsMissing(vntIterators) Then
        vntIterators = cPendingStep.Iterators
    End If

    If IsArray(vntIterators) And Not IsEmpty(vntIterators) Then
        mcLastIterator.IteratorName = cPendingStep.IteratorName
        If vntIterators(LBound(vntIterators)).IteratorType = _
                gintValue Then
            mcLastIterator.IteratorType = gintValue
            ' Since the sequence numbers begin at 0
            mcLastIterator.Sequence = gintMinIteratorSequence - 1
        Else
            mcLastIterator.IteratorType = gintFrom
            Call InitializeItRange(vntIterators,
cPendingStep.WorkspaceId, _
                    ColParameters)
        End If
    Else
        Set mcLastIterator = Nothing
    End If

    Exit Sub

InitializeItErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
        LoadResString(errIterateFailed)

End Sub

Private Sub InitializeItRange(vntIterators As Variant, ByVal lWorkspace
As Long, _
        ColParameters As cArrParameters)

    ' Initializes the LastIteration structure for range iterators from
the
    ' passed in variant containing the iterator records

    Dim lngIndex As Long
    Dim cItRec As cIterator

    On Error GoTo InitializeItRangeErr

    If IsArray(vntIterators) And Not IsEmpty(vntIterators) Then

        ' Check if the iterator range has been completely initialized
        RangeComplete (vntIterators)

        ' Initialize the Run node with the values for the From,
        ' To and Step boundaries
        For lngIndex = LBound(vntIterators) To UBound(vntIterators)
            Set cItRec = vntIterators(lngIndex)
            Select Case cItRec.IteratorType
                Case gintFrom
                    mcLastIterator.RangeFrom =
SubstituteParameters(cItRec.Value, lWorkspace,
WspParameters:=ColParameters)
                Case gintTo
```

```
                    mcLastIterator.RangeTo =
SubstituteParameters(cItRec.Value, lWorkspace,
WspParameters:=ColParameters)
                Case gintStep
                    mcLastIterator.RangeStep =
SubstituteParameters(cItRec.Value, lWorkspace,
WspParameters:=ColParameters)
                Case Else
                    On Error GoTo 0
                    Err.Raise vbObjectError + errTypeInvalid,
mstrModuleName, _
                            LoadResString(errTypeInvalid)
            End Select
        Next lngIndex

        mcLastIterator.Value = Trim$(CStr(mcLastIterator.RangeFrom -
mcLastIterator.RangeStep))
    End If

    Exit Sub

InitializeItRangeErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
        LoadResString(errIterateFailed)

End Sub
Private Function NextInSequence(vntIterators As Variant, _
        lngOldSequence As Long) As cIterator

    Dim lngIndex As Long
    Dim cItRec As cIterator

    On Error GoTo NextInSequenceErr

    If IsArray(vntIterators) And Not IsEmpty(vntIterators) Then
        For lngIndex = LBound(vntIterators) To UBound(vntIterators)
            Set cItRec = vntIterators(lngIndex)
            If cItRec.IteratorType <> gintValue Then
                On Error GoTo 0
                Err.Raise vbObjectError + errTypeInvalid, mstrModuleName,
_
                        LoadResString(errTypeInvalid)
            End If
            If cItRec.SequenceNo = lngOldSequence + 1 Then
                Exit For
            End If

        Next lngIndex

        If cItRec.SequenceNo <> lngOldSequence + 1 Then
            Set cItRec = Nothing
        End If
    Else
        Set cItRec = Nothing
    End If

    Set NextInSequence = cItRec

    Exit Function

NextInSequenceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
        LoadResString(errIterateFailed)

End Function

Public Property Get LastIterator() As cRunItDetails

    Set LastIterator = mcLastIterator

End Property
Public Property Set LastIterator(vdata As cRunItDetails)

    Set mcLastIterator = vdata

End Property

Public Property Get TasksRunning() As Integer

    TasksRunning = mintRunning

End Property

Public Property Let TasksRunning(ByVal vdata As Integer)

    mintRunning = vdata

End Property

Public Property Get TasksComplete() As Integer

    TasksComplete = mintComplete

End Property
Public Property Let TasksComplete(ByVal vdata As Integer)

    mintComplete = vdata

End Property
Public Property Get StepId() As Long

    StepId = mlngStepId

End Property
Public Property Let StepId(ByVal vdata As Long)
```

```
    mlngStepId = vdata

End Property
```

cSubSteps.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cSubSteps"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cSubSteps.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module provides a type-safe wrapper around cVector
to
'               implement a collection of cSubStep objects.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcSubSteps As cVector

Public Sub Add(ByVal objItem As cSubStep)

    mcSubSteps.Add objItem

End Sub


Public Sub Clear()

    mcSubSteps.Clear

End Sub


Public Function Count() As Long

    Count = mcSubSteps.Count

End Function


Public Function Delete(ByVal lngDelete As Long) As cSubStep

    Set Delete = mcSubSteps.Delete(lngDelete)

End Function

Public Property Get Item(ByVal Position As Long) As cSubStep
Attribute Item.VB_UserMemId = 0

    Set Item = mcSubSteps.Item(Position)

End Property

Private Sub Class_Initialize()

    Set mcSubSteps = New cVector

End Sub

Private Sub Class_Terminate()

    Set mcSubSteps = Nothing

End Sub
```

cTermProcess.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cTermProcess"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cTermProcess.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module raises an event if a completed step exists.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private WithEvents moTimer As cTimerSM
Attribute moTimer.VB_VarHelpID = -1
Private bTermProcessExists As Boolean
Public Event TermProcessExists()

Public Sub ProcessTerminated()

    bTermProcessExists = True
    moTimer.Enabled = True

End Sub
```

```
Private Sub Class_Initialize()

    bTermProcessExists = False

    Set moTimer = New cTimerSM
    moTimer.Enabled = False

End Sub

Private Sub Class_Terminate()

    Set moTimer = Nothing

End Sub

Private Sub moTimer_Timer()

    On Error GoTo moTimer_TimerErr

    If bTermProcessExists Then
        RaiseEvent TermProcessExists
    End If

    moTimer.Enabled = False
    bTermProcessExists = False

    Exit Sub

moTimer_TimerErr:
    LogErrors Errors

End Sub
```

cTermStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cTermStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cTermStep.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module encapsulates the properties of steps that
'               have completed execution such as status and time of
completion.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public TimeComplete As Currency
Public Index As Long
Public InstanceId As Long
Public ExecutionStatus As InstanceStatus
```

cTermSteps.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cTermSteps"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cTermSteps.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module provides a type-safe wrapper around cVector
to
'               implement a collection of cTermStep objects. Raises an
'               event if a step that has completed execution exists.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcTermSteps As cVector
Private WithEvents moTimer As cTimerSM
Attribute moTimer.VB_VarHelpID = -1
Public Event TermStepExists(cStepDetails As cTermStep)

Public Sub Add(ByVal citem As cTermStep)

    Call mcTermSteps.Add(citem)
    moTimer.Enabled = True

End Sub

Public Sub Clear()

    mcTermSteps.Clear

End Sub

Public Function Delete()

    Call mcTermSteps.Delete(0)
    ' Disable the timer if there are no more pending events
    If mcTermSteps.Count = 0 Then moTimer.Enabled = False

End Function
```

```
Public Property Get Item(ByVal Position As Long) As cTermStep

    Set Item = mcTermSteps(Position)

End Property

Public Function Count() As Long

    Count = mcTermSteps.Count

End Function

Private Sub Class_Initialize()

    Set mcTermSteps = New cVector

    Set moTimer = New cTimerSM
    moTimer.Enabled = False

End Sub

Private Sub Class_Terminate()

    Set mcTermSteps = Nothing
    Set moTimer = Nothing

End Sub

Private Sub moTimer_Timer()

    On Error GoTo moTimer_TimerErr

    If mcTermSteps.Count > 0 Then
        ' Since items are appended to the end of the array
        RaiseEvent TermStepExists(mcTermSteps(0))
    Else
        moTimer.Enabled = False
    End If
    Exit Sub

moTimer_TimerErr:
    LogErrors Errors

End Sub
```

cTimerSM.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cTimerSM"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cTimer.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module implements a timer.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'

Option Explicit

Public Event Timer()

Private Const mnDefaultInterval As Long = 1

Private mnTimerID As Long
Private mnInterval As Long
Private mfEnabled As Boolean

Public Property Get Interval() As Long
    Interval = mnInterval
End Property
Public Property Let Interval(Value As Long)
    If mnInterval <> Value Then
        mnInterval = Value
        If mfEnabled Then
            SetInterval mnInterval, mnTimerID
        End If
    End If
End Property

Public Property Get Enabled() As Boolean
    Enabled = mfEnabled
End Property
Public Property Let Enabled(Value As Boolean)
    If mfEnabled <> Value Then
        If Value Then
            mnTimerID = StartTimer(mnInterval)
            If mnTimerID <> 0 Then
                mfEnabled = True
                'Storing Me in the global would add a reference to Me,
which
                '   would prevent Me from being released, which in turn
would
                '   prevent my Class_Terminate code from running. To
prevent
                '   this, I store a "soft reference" - the collection
holds a
                '   pointer to me without incrementing my reference
count.
                gcTimerObjects.Add ObjPtr(Me), Str$(mnTimerID)
            End If
        Else
            StopTimer mnTimerID
```

```
            mfEnabled = False
            gcTimerObjects.Remove Str$(mnTimerID)
        End If
    End If
End Property

Private Sub Class_Initialize()
    If gcTimerObjects Is Nothing Then Set gcTimerObjects = New
Collection
    mnInterval = mnDefaultInterval
End Sub

Private Sub Class_Terminate()
    Enabled = False
End Sub

Friend Sub Tick()
    RaiseEvent Timer
End Sub
```

cVBErrorsSM.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cVBErrorsSM"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Attribute VB_Ext_KEY = "SavedWithClassBuilder" ,"Yes"
Attribute VB_Ext_KEY = "Top_Level" ,"Yes"
'  FILE:        cVBErrors.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This module encapsulates the handling of Visual Basic
errors.
'               This module does not do any error handling - any error
handler
'               will erase the errors object!
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' The Execute class exposes a method, WriteError through which we can
write to the
' error log that is currently being used by the Execute object. Store a
reference to
' Execute object locally.
Private mcExecObjRef As EXECUTEDLLLib.Execute
Public Sub WriteError(ByVal ErrorCode As errErrorConstants, _
        Optional ByVal ErrorSource As String = gstrEmptyString, _
        Optional ByVal OptArgs As String = gstrEmptyString)

    Dim sError As String

    sError = "StepMaster Error:" & ErrorCode & vbCrLf &
LoadResString(ErrorCode) & vbCrLf

    If Not StringEmpty(ErrorSource) Then
        sError = sError & "(Source: " & ErrorSource & ")" & vbCrLf
    End If
    sError = sError & OptArgs

    Call LogMessage(sError)

End Sub
Private Function InitErrorString() As String
    ' Initializes a string with all the properties of the
    ' Err object

    Dim strError As String
    Dim errCode As Long

    If Err.Number = 0 Then
        InitErrorString = gstrEmptyString
    Else
    With Err
        If Err.Number > vbObjectError And Err.Number < (vbObjectError +
65536) Then
            errCode = .Number - vbObjectError
        Else
            errCode = .Number
        End If
        strError = "Error #: " & errCode & vbCrLf
        strError = strError & "Description: " & .Description & vbCrLf
        strError = strError & "Source: " & Err.Source & vbCrLf
    End With

    Debug.Print strError
    InitErrorString = strError
    End If

End Function
Public Sub LogVBErrors()

    Dim strErr As String

    strErr = InitErrorString

    On Error GoTo LogVBErrorsErr

    If Not StringEmpty(strErr) Then
        ' Write an error using the WriteError method of the Execute
object.
        If Not mcExecObjRef Is Nothing Then
            mcExecObjRef.WriteError strErr
        Else
```

```
            WriteMessage strErr
        End If
    End If

    Err.Clear

    Exit Sub

LogVBErrorsErr:
    Call LogErrors(Errors)
    ' Since write to the error file for the step has failed, write to
the project log
    Call WriteMessage(strErr)

End Sub
Public Sub DisplayErrors()

    Dim strErr As String

    strErr = InitErrorString

    If Not StringEmpty(strErr) Then
        ' Display the error message
        MsgBox strErr
    End If

    Err.Clear

End Sub
Public Sub LogMessage(strMsg As String)

    On Error GoTo LogMessageErr

    ' Write an error using the WriteError method of the Execute object.
    If Not mcExecObjRef Is Nothing Then
        mcExecObjRef.WriteError strMsg
    Else
        WriteMessage strMsg
    End If

    Exit Sub

LogMessageErr:
    Call LogErrors(Errors)
    ' Since write to the error file for the step has failed, write to
the project log
    Call WriteMessage(strMsg)

End Sub
Public Property Set ErrorFile(vdata As EXECUTEDLLLib.Execute)

    Set mcExecObjRef = vdata

End Property
Private Sub Class_Terminate()

    Set mcExecObjRef = Nothing

End Sub
```

cVector.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cVector"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:         cVector.cls
'                Microsoft TPC-H Kit Ver. 1.00
'                Copyright Microsoft, 1999
'                All Rights Reserved
'
'
'  PURPOSE:      This class implements an array of objects.
'  Contact:      Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cVector."

' Array counter
Private mlngCount As Long
Private mcarrItems() As Object

Public Sub Add(ByVal objItem As Object)
    ' Adds the passed in Object variable to the array

    On Error GoTo AddErr

    ReDim Preserve mcarrItems(mlngCount)

    ' Set the newly added element in the array to the
    ' passed in variable
    Set mcarrItems(mlngCount) = objItem
    mlngCount = mlngCount + 1

    Exit Sub

AddErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "Add"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadInArrayFailed, _
            mstrSource, _
            LoadResString(errLoadInArrayFailed)
```

```
End Sub
Public Sub Clear()

    ' Clear the array
    ReDim mcarrItems(0)
    mlngCount = 0

End Sub

Public Function Delete(ByVal lngDelete As Long) As Object

    Dim lngIndex As Long

    On Error GoTo DeleteErr

    If lngDelete < (mlngCount - 1) Then

        ' We want to maintain the order of all items in the
        ' array - so move all remaining elements in the array
        ' up by 1
        For lngIndex = lngDelete To mlngCount - 2
            MoveDown lngIndex
        Next lngIndex

    End If

    ' Return the deleted node
    Set Delete = mcarrItems(mlngCount - 1)

    ' Delete the last Node from the array
    mlngCount = mlngCount - 1
    If mlngCount > 0 Then
        ReDim Preserve mcarrItems(0 To mlngCount - 1)
    Else
        ReDim mcarrItems(0)
    End If

    Exit Function

DeleteErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteArrayElementFailed, _
            mstrSource, _
            LoadResString(errDeleteArrayElementFailed)

End Function
Public Property Get Item(ByVal Position As Long) As Object
Attribute Item.VB_UserMemId = 0

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mlngCount Then
        Set Item = mcarrItems(Position)
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Set Item(ByVal Position As Long, _
        ByVal Value As Object)

    ' Returns the element at the passed in position in the array
    If Position >= 0 Then
        ' If the passed in position is outside the array
        ' bounds, then resize the array
        If Position >= mlngCount Then
            ReDim Preserve mcarrItems(Position)
            mlngCount = Position + 1
        End If

        ' Set the newly added element in the array to the
        ' passed in variable
        Set mcarrItems(Position) = Value
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Sub MoveUp(ByVal Position As Long)
    ' Moves the element at the passed in position up by 1

    Dim cTemp As Object

    If Position > 0 And Position < mlngCount Then
        Set cTemp = mcarrItems(Position)

        Set mcarrItems(Position) = mcarrItems(Position - 1)
        Set mcarrItems(Position - 1) = cTemp
    End If

End Sub
Public Sub MoveDown(ByVal Position As Long)
    ' Moves the element at the passed in position down by 1

    Dim cTemp As Object

    If Position >= 0 And Position < mlngCount - 1 Then
        Set cTemp = mcarrItems(Position)

        Set mcarrItems(Position) = mcarrItems(Position + 1)
        Set mcarrItems(Position + 1) = cTemp
    End If

End Sub

Public Function Count() As Long
```

```
    Count = mlngCount

End Function


Private Sub Class_Initialize()

    mlngCount = 0

End Sub


Private Sub Class_Terminate()

    Call Clear

End Sub
```

cVectorLng.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cVectorLng"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cVectorLng.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This class implements an array of longs.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cVectorLng."

' Array counter
Private mlngCount As Long
Private mcarrItems() As Long

Public Sub Add(ByVal lngItem As Long)
    ' Adds the passed in long variable to the array

    On Error GoTo AddErr

    ReDim Preserve mcarrItems(mlngCount)

    ' Set the newly added element in the array to the
    ' passed in variable
    mcarrItems(mlngCount) = lngItem
    mlngCount = mlngCount + 1

    Exit Sub

AddErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "Add"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadInArrayFailed, _
            mstrSource, _
            LoadResString(errLoadInArrayFailed)

End Sub
Public Sub Clear()

    ' Clear the array
    ReDim mcarrItems(0)

End Sub

Public Sub Delete(Optional ByVal Position As Long = -1, _
        Optional ByVal Item As Long = -1)
    ' The user can opt to delete either a specific item in
    ' the list or the item at a specified position. If no
    ' parameters are passed in, we delete the element at
    ' position 0!

    Dim lngDelete As Long
    Dim lngIndex As Long

    On Error GoTo DeleteErr

    If Position = -1 Then
        ' Since we can never store an element at position -1,
        ' we can be sure that the user is trying to delete
        ' a given item
        lngDelete = Find(Item)
    Else
        lngDelete = Position
    End If

    If lngDelete < (mlngCount - 1) Then

        ' We want to maintain the order of all items in the
        ' array - so move all remaining elements in the array
        ' up by 1
        For lngIndex = lngDelete To mlngCount - 2
            MoveDown lngIndex
        Next lngIndex

    End If
```

```
    ' Delete the last Node from the array
    mlngCount = mlngCount - 1
    If mlngCount > 0 Then
        ReDim Preserve mcarrItems(0 To mlngCount - 1)
    Else
        ReDim mcarrItems(0)
    End If

    Exit Sub

DeleteErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteArrayElementFailed, _
            mstrSource, _
            LoadResString(errDeleteArrayElementFailed)

End Sub
Public Function Find(ByVal Item As Long) As Long

    ' Returns the position at which the passed in value occurs
    ' in the array

    Dim lngIndex As Long

    On Error GoTo FindErr

    ' Find the element in the array to be deleted
    For lngIndex = 0 To mlngCount - 1

        If mcarrItems(lngIndex) = Item Then
            Find = lngIndex
            Exit Function
        End If

    Next lngIndex

    Find = -1

    Exit Function

FindErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Find"
    On Error GoTo 0
    Err.Raise vbObjectError + errItemNotFound, mstrSource, _
        LoadResString(errItemNotFound)

End Function
Public Property Get Item(ByVal Position As Long) As Long
Attribute Item.VB_UserMemId = 0

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mlngCount Then
        Item = mcarrItems(Position)
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Let Item(ByVal Position As Long, _
        ByVal Value As Long)

    ' Returns the element at the passed in position in the array
    If Position >= 0 Then
        ' If the passed in position is outside the array
        ' bounds, then resize the array
        If Position >= mlngCount Then
            ReDim Preserve mcarrItems(Position)
            mlngCount = Position + 1
        End If

        ' Set the newly added element in the array to the
        ' passed in variable
        mcarrItems(Position) = Value
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Sub MoveUp(ByVal Position As Long)
    ' Moves the element at the passed in position up by 1

    Dim lngTemp As Long

    If Position > 0 And Position < mlngCount Then
        lngTemp = mcarrItems(Position)

        mcarrItems(Position) = mcarrItems(Position - 1)
        mcarrItems(Position - 1) = lngTemp
    End If

End Sub
Public Sub MoveDown(ByVal Position As Long)
    ' Moves the element at the passed in position down by 1

    Dim lngTemp As Long

    If Position >= 0 And Position < mlngCount - 1 Then
        lngTemp = mcarrItems(Position)

        mcarrItems(Position) = mcarrItems(Position + 1)
        mcarrItems(Position + 1) = lngTemp
    End If

End Sub
```

```
Public Function Count() As Long

    Count = mlngCount

End Function


Private Sub Class_Initialize()

    mlngCount = 0

End Sub


Private Sub Class_Terminate()

    Call Clear

End Sub
```

cVectorStr.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cVectorStr"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cVectorStr.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     This class implements an array of strings.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cVectorStr."

' Array counter
Private mlngCount As Long
Private mcarrItems() As String

Public Sub Add(ByVal strItem As String)
    ' Adds the passed in string variable to the array

    On Error GoTo AddErr

    ReDim Preserve mcarrItems(mlngCount)

    ' Set the newly added element in the array to the
    ' passed in variable
    mcarrItems(mlngCount) = strItem
    mlngCount = mlngCount + 1

    Exit Sub

AddErr:
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "Add"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadInArrayFailed, _
            mstrSource, _
            LoadResString(errLoadInArrayFailed)

End Sub
Public Sub Clear()

    ' Clear the array
    ReDim mcarrItems(0)

End Sub

Public Sub Delete(Optional ByVal Position As Long = -1, _
        Optional ByVal Item As String = -1)
    ' The user can opt to delete either a specific item in
    ' the list or the item at a specified position. If no
    ' parameters are passed in, we delete the element at
    ' position 0!

    Dim lngDelete As Long
    Dim lngIndex As Long

    On Error GoTo DeleteErr
    mstrSource = mstrModuleName & "Delete"

    If Position = -1 Then
        ' Since we can never store an element at position -1,
        ' we can be sure that the user is trying to delete
        ' a given item
        lngDelete = Find(Item)
    Else
        lngDelete = Position
    End If

    If lngDelete < (mlngCount - 1) Then

        ' We want to maintain the order of all items in the
        ' array - so move all remaining elements in the array
        ' up by 1
        For lngIndex = lngDelete To mlngCount - 2
            MoveDown lngIndex
        Next lngIndex
```

```
    End If

    ' Delete the last Node from the array
    mlngCount = mlngCount - 1
    If mlngCount > 0 Then
        ReDim Preserve mcarrItems(0 To mlngCount - 1)
    Else
        ReDim mcarrItems(0)
    End If

    Exit Sub

DeleteErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteArrayElementFailed, _
            mstrSource, _
            LoadResString(errDeleteArrayElementFailed)

End Sub
Public Function Find(ByVal Item As String) As Long

    ' Returns the position at which the passed in value occurs
    ' in the array

    Dim lngIndex As Long

    On Error GoTo FindErr
    mstrSource = mstrModuleName & "Find"

    ' Find the element in the array to be deleted
    For lngIndex = 0 To mlngCount - 1

        If mcarrItems(lngIndex) = Item Then
            Find = lngIndex
            Exit Function
        End If

    Next lngIndex

    Find = -1

    Exit Function

FindErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "Find"
    On Error GoTo 0
    Err.Raise vbObjectError + errItemNotFound, mstrSource, _
        LoadResString(errItemNotFound)

End Function
Public Property Get Item(ByVal Position As Long) As String
Attribute Item.VB_UserMemId = 0

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mlngCount Then
        Item = mcarrItems(Position)
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Let Item(ByVal Position As Long, _
        ByVal Value As String)

    ' Returns the element at the passed in position in the array
    If Position >= 0 Then
        ' If the passed in position is outside the array
        ' bounds, then resize the array
        If Position >= mlngCount Then
            ReDim Preserve mcarrItems(Position)
            mlngCount = Position + 1
        End If

        ' Set the newly added element in the array to the
        ' passed in variable
        mcarrItems(Position) = Value
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Sub MoveUp(ByVal Position As Long)
    ' Moves the element at the passed in position up by 1

    Dim strTemp As String

    If Position > 0 And Position < mlngCount Then
        strTemp = mcarrItems(Position)

        mcarrItems(Position) = mcarrItems(Position - 1)
        mcarrItems(Position - 1) = strTemp
    End If

End Sub
Public Sub MoveDown(ByVal Position As Long)
    ' Moves the element at the passed in position down by 1

    Dim strTemp As String

    If Position >= 0 And Position < mlngCount - 1 Then
        strTemp = mcarrItems(Position)

        mcarrItems(Position) = mcarrItems(Position + 1)
        mcarrItems(Position + 1) = strTemp
    End If
```

```
End Sub

Public Function Count() As Long

    Count = mlngCount

End Function


Private Sub Class_Initialize()

    mlngCount = 0

End Sub
Private Sub Class_Terminate()

    Call Clear

End Sub
```

cWorker.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cWorker"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cWorker.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a worker step.
'               Implements the cStep class - carries out initializations
'               and validations that are specific to worker steps.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Implements cStep

' Object variable to keep the step reference in
Private mcStep As cStep

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cWorker."
Private Sub cStep_AddAllIterators()

    Call mcStep.AddAllIterators

End Sub

Private Property Let cStep_StartDir(ByVal RHS As String)

    mcStep.StartDir = RHS

End Property

Private Property Get cStep_StartDir() As String

    cStep_StartDir = mcStep.StartDir

End Property

Private Property Set cStep_NodeDB(RHS As DAO.Database)

    Set mcStep.NodeDB = RHS

End Property

Private Property Get cStep_NodeDB() As DAO.Database

    Set cStep_NodeDB = mcStep.NodeDB

End Property

Private Function cStep_IncVersionY() As String

    cStep_IncVersionY = mcStep.IncVersionY

End Function
Private Function cStep_IsNewVersion() As Boolean
    cStep_IsNewVersion = mcStep.IsNewVersion
End Function
Private Function cStep_OldVersionNo() As String
    cStep_OldVersionNo = mcStep.OldVersionNo
End Function

Private Function cStep_IncVersionX() As String

    cStep_IncVersionX = mcStep.IncVersionX

End Function
Private Sub cStep_UpdateIteratorVersion()

    Call mcStep.UpdateIteratorVersion

End Sub

Private Function cStep_IteratorCount() As Long

    cStep_IteratorCount = mcStep.IteratorCount
```

```
End Function

Private Sub cStep_UnloadIterators()

    Call mcStep.UnloadIterators

End Sub

Private Sub cStep_SaveIterators()

    Call mcStep.SaveIterators

End Sub
Private Property Get cStep_IteratorName() As String

    cStep_IteratorName = mcStep.IteratorName

End Property
Private Property Let cStep_IteratorName(ByVal RHS As String)

    mcStep.IteratorName = RHS

End Property

Private Sub cStep_LoadIterator(cItRecord As cIterator)

    Call mcStep.LoadIterator(cItRecord)

End Sub
Private Sub cStep_DeleteIterator(cItRecord As cIterator)

    Call mcStep.DeleteIterator(cItRecord)

End Sub

Private Sub cStep_InsertIterator(cItRecord As cIterator)

    Call mcStep.InsertIterator(cItRecord)

End Sub
Private Function cStep_Iterators() As Variant

    cStep_Iterators = mcStep.Iterators

End Function
Private Sub cStep_ModifyIterator(cItRecord As cIterator)

    Call mcStep.ModifyIterator(cItRecord)

End Sub
Private Sub cStep_RemoveIterator(cItRecord As cIterator)

    Call mcStep.RemoveIterator(cItRecord)

End Sub
Private Sub cStep_UpdateIterator(cItRecord As cIterator)

    Call mcStep.UpdateIterator(cItRecord)

End Sub
Private Sub cStep_AddIterator(cItRecord As cIterator)

    Call mcStep.AddIterator(cItRecord)

End Sub

Private Property Let cStep_Position(ByVal RHS As Long)

    mcStep.Position = RHS

End Property

Private Property Get cStep_Position() As Long

    cStep_Position = mcStep.Position

End Property

Private Function cStep_Clone(Optional cCloneStep As cStep) As cStep

    Dim cNewWorker As cWorker

    Set cNewWorker = New cWorker
    Set cStep_Clone = mcStep.Clone(cNewWorker)

End Function

Private Sub StepTextOrFileEntered()
    ' Checks if either the step text or the name of the file containing
    ' the text has been entered
    ' If both of them are null or both of them are not null,
    ' the worker step is invalid and an error is raised
    If StringEmpty(mcStep.StepText) And StringEmpty(mcStep.StepTextFile)
Then
        ShowError errStepTextAndFileNull
        On Error GoTo 0
        Err.Raise vbObjectError + errStepTextAndFileNull, _
            mstrSource, LoadResString(errStepTextAndFileNull)
    End If

End Sub

Private Property Get cStep_IndOperation() As Operation

    cStep_IndOperation = mcStep.IndOperation

End Property

Private Property Let cStep_IndOperation(ByVal RHS As Operation)

    mcStep.IndOperation = RHS
```

```
End Property

Private Property Get cStep_NextStepId() As Long

    cStep_NextStepId = mcStep.NextStepId

End Property

Private Property Let cStep_OutputFile(ByVal RHS As String)

    mcStep.OutputFile = RHS

End Property

Private Property Get cStep_OutputFile() As String

    cStep_OutputFile = mcStep.OutputFile

End Property

Private Property Let cStep_ErrorFile(ByVal RHS As String)

    mcStep.ErrorFile = RHS

End Property

Private Property Get cStep_ErrorFile() As String

    cStep_ErrorFile = mcStep.ErrorFile

End Property
'Private Property Let cStep_LogFile(ByVal RHS As String)
'
'    mcStep.LogFile = RHS
'
'End Property
'
'Private Property Get cStep_LogFile() As String
'
'    cStep_LogFile = mcStep.LogFile
'
'End Property

Private Property Let cStep_ArchivedFlag(ByVal RHS As Boolean)

    mcStep.ArchivedFlag = RHS

End Property

Private Property Get cStep_ArchivedFlag() As Boolean

    cStep_ArchivedFlag = mcStep.ArchivedFlag

End Property

Private Sub Class_Initialize()

    ' Create the object
    Set mcStep = New cStep

    ' Initialize the object with valid values for a Worker step
    ' The global flag should be the first field to be initialized
    ' since subsequent validations might try to check if the
    ' step being created is global
    mcStep.GlobalFlag = False
'    mcStep.GlobalRunMethod = gintNoOption
    mcStep.StepType = gintWorkerStep

End Sub
Private Sub Class_Terminate()

    ' Remove the step object
    Set mcStep = Nothing

End Sub
Private Sub cStep_Add()

    ' Call a private procedure to see if the step text has been
    ' entered - since a worker step actually executes a step, entry
    ' of the text is mandatory
    Call StepTextOrFileEntered

    ' Call the Add method of the step class to carry out the insert
    mcStep.Add

End Sub

Private Property Get cStep_ContinuationCriteria() As
ContinuationCriteria

    cStep_ContinuationCriteria = mcStep.ContinuationCriteria

End Property

Private Property Let cStep_ContinuationCriteria(ByVal RHS As
ContinuationCriteria)

    ' The Continuation criteria must be non-null for all worker steps.
    ' Check if the Continuation Criteria is valid
    Select Case RHS
        Case gintOnFailureAbortSiblings, gintOnFailureCompleteSiblings,
_
                gintOnFailureSkipSiblings, gintOnFailureAbort, _
                gintOnFailureContinue, gintOnFailureAsk
            mcStep.ContinuationCriteria = RHS

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errContCriteriaInvalid, _
                    mstrModuleName,
LoadResString(errContCriteriaInvalid)
    End Select
```

```
End Property

Private Property Let cStep_DegreeParallelism(ByVal RHS As String)

    mcStep.DegreeParallelism = RHS

End Property

Private Property Get cStep_DegreeParallelism() As String

    cStep_DegreeParallelism = mcStep.DegreeParallelism

End Property

Private Sub cStep_Delete()

    mcStep.Delete

End Sub

Private Property Get cStep_EnabledFlag() As Boolean

    cStep_EnabledFlag = mcStep.EnabledFlag

End Property

Private Property Let cStep_EnabledFlag(ByVal RHS As Boolean)

    mcStep.EnabledFlag = RHS

End Property

Private Property Let cStep_ExecutionMechanism(ByVal RHS As
ExecutionMethod)

    On Error GoTo ExecutionMechanismErr
    mstrSource = mstrModuleName & "cStep_ExecutionMechanism"

    Select Case RHS
        Case gintExecuteShell, gintExecuteODBC
            mcStep.ExecutionMechanism = RHS

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errExecutionMechanismInvalid, _
                    mstrSource,
LoadResString(errExecutionMechanismInvalid)
    End Select

    Exit Property

ExecutionMechanismErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "cStep_ExecutionMechanism"
    On Error GoTo 0
    Err.Raise vbObjectError + errExecutionMechanismLetFailed, _
        mstrSource, LoadResString(errExecutionMechanismLetFailed)

End Property

Private Property Get cStep_ExecutionMechanism() As ExecutionMethod

    cStep_ExecutionMechanism = mcStep.ExecutionMechanism

End Property

Private Property Let cStep_FailureDetails(ByVal RHS As String)

    mcStep.FailureDetails = RHS

End Property

Private Property Get cStep_FailureDetails() As String

    cStep_FailureDetails = mcStep.FailureDetails

End Property

Private Property Get cStep_GlobalFlag() As Boolean

    cStep_GlobalFlag = mcStep.GlobalFlag

End Property

Private Property Let cStep_GlobalFlag(ByVal RHS As Boolean)

    ' Set the global flag to false - this flag is initialized when
    ' an instance of the class is created. Just making sure that
    ' nobody changes the value inadvertently
    mcStep.GlobalFlag = False

End Property
Private Sub cStep_Modify()

    ' Call a private procedure to see if the step text has been
    ' entered - since a worker step actually executes a step, entry
    ' of the text is mandatory
    Call StepTextOrFileEntered

    ' Call the Modify method of the step class to carry out the update
    mcStep.Modify

End Sub

Private Property Let cStep_ParentStepId(ByVal RHS As Long)

    mcStep.ParentStepId = RHS

End Property

Private Property Get cStep_ParentStepId() As Long
```

```
    cStep_ParentStepId = mcStep.ParentStepId
End Property
Private Property Let cStep_ParentVersionNo(ByVal RHS As String)
    mcStep.ParentVersionNo = RHS
End Property
Private Property Get cStep_ParentVersionNo() As String
    cStep_ParentVersionNo = mcStep.ParentVersionNo
End Property
Private Property Let cStep_SequenceNo(ByVal RHS As Integer)
    mcStep.SequenceNo = RHS
End Property
Private Property Get cStep_SequenceNo() As Integer
    cStep_SequenceNo = mcStep.SequenceNo
End Property
Private Property Let cStep_StepId(ByVal RHS As Long)
    mcStep.StepId = RHS
End Property
Private Property Get cStep_StepId() As Long
    cStep_StepId = mcStep.StepId
End Property

Private Property Let cStep_StepLabel(ByVal RHS As String)
    mcStep.StepLabel = RHS
End Property
Private Property Get cStep_StepLabel() As String
    cStep_StepLabel = mcStep.StepLabel
End Property

Private Property Let cStep_StepLevel(ByVal RHS As Integer)
    mcStep.StepLevel = RHS
End Property
Private Property Get cStep_StepLevel() As Integer
    cStep_StepLevel = mcStep.StepLevel
End Property
Private Property Let cStep_StepText(ByVal RHS As String)
    mcStep.StepText = RHS
End Property
Private Property Get cStep_StepText() As String
    cStep_StepText = mcStep.StepText
End Property
Private Property Let cStep_StepTextFile(ByVal RHS As String)
    mcStep.StepTextFile = RHS
End Property
Private Property Get cStep_StepTextFile() As String
    cStep_StepTextFile = mcStep.StepTextFile
End Property
Private Property Let cStep_StepType(RHS As gintStepType)
    mcStep.StepType = gintWorkerStep
End Property
Private Property Get cStep_StepType() As gintStepType
    cStep_StepType = mcStep.StepType
End Property
Private Sub cStep_Validate()
    ' The validate routines for each of the steps will
    ' carry out the specific validations for the type and
    ' call the generic validation routine

    On Error GoTo cStep_ValidateErr

    ' Validations specific to worker steps
```

```
    ' Check if the step text or a file name has been
    ' specified
    Call StepTextOrFileEntered

    mcStep.Validate

    Exit Sub

cStep_ValidateErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "cStep_Validate"
    On Error GoTo 0
    Err.Raise vbObjectError + errValidateFailed, _
            mstrSource, _
            LoadResString(errValidateFailed)
End Sub

Private Property Let cStep_VersionNo(ByVal RHS As String)

    mcStep.VersionNo = RHS

End Property

Private Property Get cStep_VersionNo() As String

    cStep_VersionNo = mcStep.VersionNo

End Property

Private Property Let cStep_WorkspaceId(ByVal RHS As Long)

    mcStep.WorkspaceId = RHS

End Property

Private Property Get cStep_WorkspaceId() As Long

    cStep_WorkspaceId = mcStep.WorkspaceId

End Property
```

cWorkspace.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cWorkspace"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:        cWorkspace.cls
'               Microsoft TPC-H Kit Ver. 1.00
'               Copyright Microsoft, 1999
'               All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a workspace.
'               Contains functions to insert, update and delete
'               att_workspaces records from the database.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Local variable(s) to hold property value(s)
Private mlngWorkspaceId As Long
Private mstrWorkspaceName As String
Private mblnArchivedFlag As Boolean
Private mdbsStepMaster As Database

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cWorkspace."

' The cSequence class is used to generate unique workspace identifiers
Private mWorkspaceSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM

Public Function Clone() As cWorkspace

    ' Creates a copy of a given workspace

    Dim cCloneWsp As cWorkspace

    On Error GoTo CloneErr

    Set cCloneWsp = New cWorkspace

    ' Copy all the workspace properties to the newly
    ' created workspace
    cCloneWsp.WorkspaceId = mlngWorkspaceId
    cCloneWsp.WorkspaceName = mstrWorkspaceName
    cCloneWsp.ArchivedFlag = mblnArchivedFlag

    ' And set the return value to the newly created workspace
    Set Clone = cCloneWsp

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, _
            mstrSource, LoadResString(errCloneFailed)

End Function
```

```
Public Property Let ArchivedFlag(ByVal vdata As Boolean)

    mblnArchivedFlag = vdata

End Property

Public Property Get ArchivedFlag() As Boolean

    ArchivedFlag = mblnArchivedFlag

End Property

Public Property Set WorkDatabase(vdata As Database)

    Set mdbsStepMaster = vdata

End Property

Private Sub WorkspaceNameDuplicate()
    ' Check if the workspace name already exists in the workspace

    Dim rstWorkspace As Recordset
    Dim strSql As String
    Dim qy As DAO.QueryDef

    On Error GoTo WorkspaceNameDuplicateErr
    mstrSource = mstrModuleName & "WorkspaceNameDuplicate"

    ' Create a recordset to retrieve the count of records
    ' having the same workspace name
    strSql = " Select count(*) as workspace_count " & _
        " from att_workspaces " & _
        " where workspace_name = [w_name] " & _
        " and workspace_id <> [w_id] "
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strSql)

    ' Call a procedure to assign the parameter values
    Call AssignParameters(qy)

    Set rstWorkspace = qy.OpenRecordset(dbOpenForwardOnly)

'        mFieldValue.MakeStringFieldValid (mstrWorkspaceName) & _
'        " and workspace_id <> " & _
'        Str(mlngWorkspaceId)
'
'    Set rstWorkspace = mdbsStepMaster.OpenRecordset( _
'        strSQL, dbOpenForwardOnly)

    If rstWorkspace![workspace_count] > 0 Then
        rstWorkspace.Close
        qy.Close
        ShowError errDuplicateWorkspaceName
        On Error GoTo 0
        Err.Raise vbObjectError + errDuplicateWorkspaceName, _
            mstrSource, LoadResString(errDuplicateWorkspaceName)
    End If
    rstWorkspace.Close
    qy.Close

    Exit Sub

WorkspaceNameDuplicateErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "WorkspaceNameDuplicate"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceNameDuplicateFailed, _
        mstrSource, LoadResString(errWorkspaceNameDuplicateFailed)

End Sub
Public Property Let WorkspaceName(vdata As String)

    On Error GoTo WorkspaceNameErr
    mstrSource = mstrModuleName & "WorkspaceName"

    If vdata = gstrEmptyString Then

        On Error GoTo 0
        ' Propogate this error back to the caller
        Err.Raise vbObjectError + errWorkspaceNameMandatory, _
            mstrSource, LoadResString(errWorkspaceNameMandatory)
    Else
        mstrWorkspaceName = vdata
    End If
    Exit Property

WorkspaceNameErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "WorkspaceName"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceNameSetFailed, _
        mstrSource, LoadResString(errWorkspaceNameSetFailed)

End Property

Public Property Let WorkspaceId(vdata As Long)

    On Error GoTo WorkspaceIdErr
    mstrSource = mstrModuleName & "WorkspaceId"

    If (vdata > 0) Then
        mlngWorkspaceId = vdata
    Else
        ' Propogate this error back to the caller
        On Error GoTo 0
        Err.Raise vbObjectError + errWorkspaceIdInvalid, _
            mstrSource, LoadResString(errWorkspaceIdInvalid)
    End If

    Exit Property

WorkspaceIdErr:
    LogErrors Errors
```

```
    mstrSource = mstrModuleName & "WorkspaceId"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceIdSetFailed, _
        mstrSource, LoadResString(errWorkspaceIdSetFailed)

End Property

Public Sub AddWorkspace()

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddWorkspaceErr

    ' Retrieve the next identifier using the sequence class
    Set mWorkspaceSeq = New cSequence
    Set mWorkspaceSeq.IdDatabase = mdbsStepMaster
    mWorkspaceSeq.IdentifierColumn = FLD_ID_WORKSPACE
    mlngWorkspaceId = mWorkspaceSeq.Identifier
    Set mWorkspaceSeq = Nothing

    ' Call procedure to raise an error if the Workspace name
    ' already exists in the db
    Call WorkspaceNameDuplicate

    ' A new record will have the archived_flag turned off
    mblnArchivedFlag = False

    ' Create a temporary querydef object
    strInsert = "insert into att_workspaces " & _
            "( workspace_id, workspace_name, " & _
            " archived_flag ) " & _
            " values ( [w_id], [w_name], [archived] ) "
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to assign the parameter values
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close

'    strInsert = "insert into att_workspaces " & _
'            "( workspace_id, workspace_name, " & _
'            " archived_flag ) " & _
'            " values ( " & _
'            Str(mlngWorkspaceId) & _
'            ", " & mFieldValue.MakeStringFieldValid(mstrWorkspaceName)
& _
'            ", " & Str(mblnArchivedFlag) & _
'            " ) "
'    mdbsStepMaster.Execute strInsert, dbFailOnError
'
    Exit Sub

AddWorkspaceErr:

    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "AddWorkspace"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceInsertFailed, _
        mstrSource, LoadResString(errWorkspaceInsertFailed)

End Sub
Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the field names. When the parameter names are
    ' the same as the field names, parameters in the where
    ' clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr
    mstrSource = mstrModuleName & "AssignParameters"

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[w_id]"
                prmParam.Value = mlngWorkspaceId

            Case "[w_name]"
                prmParam.Value = mstrWorkspaceName

            Case "[archived]"
                prmParam.Value = mblnArchivedFlag

            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource, _
                        prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrSource, _
                        LoadResString(errInvalidParameter)
        End Select
    Next prmParam

    Exit Sub

AssignParametersErr:

    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrSource, LoadResString(errAssignParametersFailed)

End Sub
Public Sub DeleteWorkspace()

    Dim strDelete As String
    Dim qy As DAO.QueryDef
```

```
    On Error GoTo DeleteWorkspaceErr

    strDelete = "delete from att_workspaces " & _
        " where workspace_id = [w_id]"
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strDelete)

    ' Call a procedure to assign the parameter values
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close

'    mdbsStepMaster.Execute strDelete, dbFailOnError
'        " where workspace_id = " & _
'        Str(mlngWorkspaceId)

    Exit Sub

DeleteWorkspaceErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "DeleteWorkspace"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceDeleteFailed, _
        mstrSource, LoadResString(errWorkspaceDeleteFailed)
End Sub

Public Sub ModifyWorkspace()

    Dim strUpdate As String
    Dim qy As DAO.QueryDef

    On Error GoTo ModifyWorkspaceErr

    ' Call procedure to raise an error if the Workspace name
    ' already exists in the db
    Call WorkspaceNameDuplicate

    strUpdate = "update att_workspaces " & _
        " set workspace_name = [w_name] " & _
        ", archived_flag = [archived] " & _
        " where workspace_id = [w_id] "
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the parameter values
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close

'    strUpdate = "update att_workspaces " & _
'        " set workspace_name = " & _
'        mFieldValue.MakeStringFieldValid(mstrWorkspaceName) & _
'        ", archived_flag = " & _
'        Str(mblnArchivedFlag) & _
'        " where workspace_id = " & _
'        Str(mlngWorkspaceId)
'
'    mdbsStepMaster.Execute strUpdate, dbFailOnError
'
    Exit Sub

ModifyWorkspaceErr:

    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "ModifyWorkspace"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceUpdateFailed, _
        mstrSource, LoadResString(errWorkspaceUpdateFailed)

End Sub
Public Property Get WorkspaceName() As String

    WorkspaceName = mstrWorkspaceName

End Property

Public Property Get WorkspaceId() As Long

    WorkspaceId = mlngWorkspaceId

End Property

Private Sub Class_Initialize()

    ' Each function will append it's own name to this
    ' variable
    mstrSource = "cWorkspace."

    Set mFieldValue = New cStringSM

End Sub

Private Sub Class_Terminate()

    Set mdbsStepMaster = Nothing
    Set mFieldValue = Nothing

End Sub
```

DatabaseSM.bas

```
Attribute VB_Name = "DatabaseSM"
'  FILE:       DatabaseSM.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains all the database initialization/cleanup
'              procedures for the project. Also contains upgrade
'              database upgrade functions.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
```

```
'
' This module is called DatabaseSM, since Database is a standard
' Visual Basic object and we want to avoid any confusion with it.

Option Explicit

Public wrkJet As Workspace
Public dbsAttTool As Database
Public gblnDbOpen As Boolean
Public gRunEngine As rdoEngine

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "DatabaseSM."
Public Const gsDefDBFileExt As String = ".stp"
Private Const msDefDBFile As String = "\SMData" & gsDefDBFileExt

Private Const merrFileNotFound As Integer = 3024
Private Const merrDaoTableMissing As Integer = 3078

Private Const STEPMASTER_SETTINGS_VAL_NAME_DBFILE As String =
"WorkspaceFile"

Public Const DEF_NO_COUNT_DISPLAY          As Boolean = False
Public Const DEF_NO_EXECUTE                As Boolean = False
Public Const DEF_PARSE_QUERY_ONLY          As Boolean = False
Public Const DEF_ANSI_QUOTED_IDENTIFIERS   As Boolean = False
Public Const DEF_ANSI_NULLS                As Boolean = True
Public Const DEF_SHOW_QUERY_PLAN           As Boolean = False
Public Const DEF_SHOW_STATS_TIME           As Boolean = False
Public Const DEF_SHOW_STATS_IO             As Boolean = False
Public Const DEF_PARSE_ODBC_MSG_PREFIXES   As Boolean = True
Public Const DEF_ROW_COUNT                 As Long = 0
Public Const DEF_TSQL_BATCH_SEPARATOR      As String = "GO"
Public Const DEF_QUERY_TIME_OUT            As Long = 0
Public Const DEF_SERVER_LANGUAGE           As String = "(Default)"
Public Const DEF_CHARACTER_TRANSLATION     As Boolean = True
Public Const DEF_REGIONAL_SETTINGS         As Boolean = False

Public Const PARAM_DEFAULT_DIR             As String = "DEFAULT_DIR"
Public Const PARAM_DEFAULT_DIR_DESC        As String = "Default
destination directory " & _
               "for all output and error files. If it is blank, the
StepMaster installation directory will be used."

Public Const PARAM_RUN_ID                  As String = "RUN_ID"
Public Const PARAM_RUN_ID_DESC             As String = "The run
identifier for a run. " & _
               "Any modifications will be overwritten before each run."

Public Const PARAM_OUTPUT_DIR              As String = "OUTPUT_DIR"
Public Const PARAM_OUTPUT_DIR_DESC         As String = "The output
directory for a run. " & _
               "Any modifications will be overwritten before each run."

Public Const CONNECTION_STRINGS_TO_NAME_SUFFIX As String = "_NAME"

Private Const TBL_RUN_STEP_HDR As String = "run_header"
Private Const TBL_RUN_STEP_DTLS As String = "run_step_details"
Public Const TBL_CONNECTION_DTLS As String = "connection_dtls"
Public Const TBL_CONNECTION_STRINGS As String = "workspace_connections"
Public Const TBL_STEPS As String = "att_steps"

Public Const FLD_ID_CONN_NAME As String = "connection_name_id"
Public Const FLD_ID_WORKSPACE As String = "workspace_id"
Public Const FLD_ID_STEP As String = "step_id"
Public Const FLD_ID_PARAMETER As String = "parameter_id"

Public Const FLD_CONN_DTL_CONNECTION_NAME As String = "connection_name"
Public Const FLD_CONN_DTL_CONNECTION_STRING As String =
"connection_string_name"
Public Const FLD_CONN_DTL_CONNECTION_TYPE As String = "connection_type"

Public Const FLD_CONN_STR_CONNECTION_NAME As String = "connection_name"

Public Const FLD_STEPS_EXEC_MECHANISM As String = "execution_mechanism"
Public Const FLD_STEPS_EXEC_DTL As String = "start_directory"
Public Const FLD_STEPS_VERSION_NO As String = "version_no"

Public Const DATA_TYPE_CURRENCY As String = "CURRENCY"
Public Const DATA_TYPE_LONG As String = "Long"
Public Const DATA_TYPE_INTEGER As String = "INTEGER"
Public Const DATA_TYPE_TEXT255 As String = "Text(255)"

Private Sub InsertBuiltInParameter(dbFile As Database, sParamName As
String, _
        sParamValue As String, sParamDesc As String)

    Dim sBuf As String
    Dim cTempStr As New cStringSM
    Dim lId As Long
    Dim rTemp As DAO.Recordset
    Dim rParam As DAO.Recordset
    Dim cTempSeq As cSequence

    ' Create the passed in built-in parameter, for each workspace in the
db
    Set cTempSeq = New cSequence
    Set cTempSeq.IdDatabase = dbFile
    cTempSeq.IdentifierColumn = FLD_ID_PARAMETER

    sBuf = "select * from att_workspaces "
    Set rTemp = dbFile.OpenRecordset(sBuf, dbOpenSnapshot)
    If rTemp.RecordCount <> 0 Then
        rTemp.MoveFirst

        While Not rTemp.EOF
            sBuf = "select * from workspace_parameters " & _
                    " where workspace_id = " & Str(rTemp!workspace_id) &
_
                    " and parameter_name = " &
cTempStr.MakeStringFieldValid(sParamName)
```

```
            Set rParam = dbFile.OpenRecordset(sBuf, dbOpenSnapshot)
            If rParam.RecordCount <> 0 Then
                rParam.MoveFirst
                ' Since the parameter already exists, change it to a
built-in type
                sBuf = "update workspace_parameters " & _
                       " set parameter_type = " &
CStr(gintParameterBuiltIn) & _
                       ", description = " &
cTempStr.MakeStringFieldValid(sParamDesc) & _
                       " where workspace_id = " &
Str(rTemp!workspace_id) & _
                       " and parameter_id = " &
Str(rParam!parameter_id)
            Else
                ' Else, insert a parameter record
                lId = cTempSeq.Identifier
                sBuf = "insert into workspace_parameters " & _
                       "( workspace_id, parameter_id, " & _
                       " parameter_name, parameter_value, " & _
                       " description, parameter_type ) " & _
                       " values ( " & _
                       Str(rTemp!workspace_id) & ", " & Str(lId) & ", "
& _
                       cTempStr.MakeStringFieldValid(sParamName) & ", "
& _
                       cTempStr.MakeStringFieldValid(sParamValue) & ",
" & _
                       cTempStr.MakeStringFieldValid(sParamDesc) & ", "
& _
                       CStr(gintParameterBuiltIn) & _
                       " ) "
            End If
            dbFile.Execute sBuf, dbFailOnError
            rParam.Close

            rTemp.MoveNext
        Wend
    End If
    rTemp.Close

End Sub
Public Sub InitRunEngine()

    Set gRunEngine = New rdoEngine
    gRunEngine.rdoDefaultCursorDriver = rdUseServer

End Sub

Public Function DefaultDBFile() As String
    DefaultDBFile = GetSetting(App.Title, "Settings",
STEPMASTER_SETTINGS_VAL_NAME_DBFILE, App.Path & msDefDBFile)
End Function

Public Sub CloseDatabase()
    Dim dbsInstance As Database
    Dim recInstance As Recordset

    On Error GoTo CloseDatabaseErr

    ' Close all open recordsets and databases in the workspace
    For Each dbsInstance In wrkJet.Databases

        For Each recInstance In dbsAttTool.Recordsets
            recInstance.Close
        Next recInstance
        dbsInstance.Close

    Next dbsInstance

    Set dbsAttTool = Nothing

    gblnDbOpen = False
    wrkJet.Close

    Exit Sub

CloseDatabaseErr:

    Call LogErrors(Errors)
    Resume Next

End Sub

Private Function NoDbChanges(sVerTo As String, sVerFrom As String) As
Boolean

    If sVerTo = gsVersion242 And sVerFrom = gsVersion241 Then
        NoDbChanges = True
    ElseIf sVerTo = gsVersion242 And sVerFrom = gsVersion24 Then
        NoDbChanges = True
    ElseIf sVerTo = gsVersion253 And sVerFrom = gsVersion251 Then
        NoDbChanges = True
    ElseIf sVerTo = gsVersion255 And sVerFrom = gsVersion251 Then
        NoDbChanges = True
    Else
        NoDbChanges = False
    End If

End Function

Public Function SMOpenDatabase(Optional strDbName As String =
gstrEmptyString) As Boolean
    Dim sVersion As String
    Dim bOpeningDb As Boolean ' This flag is used to check if
OpenDatabase failed

    On Error GoTo OpenDatabaseErr

    bOpeningDb = False
    SMOpenDatabase = False
```

```
    ' Create Microsoft Jet Workspace object.
    If Not gblnDbOpen Then
        Set wrkJet = CreateWorkspace("att_tool_workspace_setup", "admin",
gstrEmptyString, dbUseJet)
    End If

    ' Prompt the user for the database file if it is not passed in
    If StringEmpty(strDbName) Then
        strDbName = BrowseDBFile
        If StringEmpty(strDbName) Then
            Exit Function
        End If
    End If

    Do
        If gblnDbOpen Then
#If Not RUN_ONLY Then
            CloseOpenWorkspaces
#End If
            Set wrkJet = CreateWorkspace("att_tool_workspace_setup",
"admin", gstrEmptyString, dbUseJet)
        End If

        ' Toggle the bOpeningDb flag around the OpenDatabase method -
the value
        ' of this flag will be checked by the error handler to determine
if it is
        ' the OpenDatabase that failed.
        BugMessage "DB File: " & strDbName

        bOpeningDb = True
        ' Open the database for exclusive use
        Set dbsAttTool = wrkJet.OpenDatabase(strDbName, Options:=True)
        bOpeningDb = False

        If dbsAttTool Is Nothing Then
            ' If the file is not present in the directory, display
            ' an error and ask the user to enter a new path
            Call ShowError(errOpenDbFailed, OptArgs:=strDbName)

            strDbName = BrowseDBFile
        Else
            sVersion = DBVersion(dbsAttTool)

            ' Make sure the application and db version numbers match
            If sVersion = gsVersion Then
                Call InitializeData(strDbName)
                gblnDbOpen = True
                SMOpenDatabase = True
            Else
                If UpgradeDb(wrkJet, dbsAttTool, gsVersion, sVersion)
Then
                    Call InitializeData(strDbName)
                    gblnDbOpen = True
                    SMOpenDatabase = True
                Else
                    dbsAttTool.Close
                    Set dbsAttTool = Nothing

                    ShowError errVersionMismatch, _
                            OptArgs:=" Please install Version '" &
gsVersion & "' of the workspace definition file."
                    strDbName = BrowseDBFile
                End If
            End If
        End If
    Loop While gblnDbOpen = False And Not StringEmpty(strDbName)

    Exit Function

OpenDatabaseErr:
    Call DisplayErrors(Errors)

    ' If the OpenDatabase failed, continue
    If bOpeningDb Then
        Resume Next
    End If

    Call ShowError(errOpenDbFailed, OptArgs:=strDbName)

End Function
Private Sub InitializeData(sDb As String)

    Set gcParameters = New cArrParameters
    Set gcParameters.ParamDatabase = dbsAttTool

    Set gcSteps = New cArrSteps
    Set gcSteps.StepDB = dbsAttTool

    Set gcConstraints = New cArrConstraints
    Set gcConstraints.ConstraintDB = dbsAttTool

    Set gcConnections = New cConnections
    Set gcConnections.ConnDb = dbsAttTool

    Set gcConnDtls = New cConnDtls
    Set gcConnDtls.ConnDb = dbsAttTool

    ' Disable the error handler since this is not a critical step
    On Error GoTo 0
    SaveSetting App.Title, "Settings",
STEPMASTER_SETTINGS_VAL_NAME_DBFILE, sDb
End Sub
Private Sub UpdateContinuationCriteria(dbFile As DAO.Database)

    Dim qyTemp As DAO.QueryDef
    Dim sBuf As String

    On Error GoTo UpdateContinuationCriteriaErr
```

```
    sBuf = "Since this version of the executable incorporates failure
processing, " & _
           "the upgrade will update the On Failure field for each of
the steps " & _
           "to 'Continue' to be compatible with the existing behaviour.
" & _
           "Proceed?"
    If Not Confirm(Buttons:=vbYesNo, strMessage:=sBuf,
strTitle:="Upgrade database") Then
        Exit Sub
    End If

    ' Create a recordset object to retrieve all steps for
    ' the given workspace
    sBuf = " update att_steps a " & _
        " set continuation_criteria = " & CStr(gintOnFailureContinue) &
_
        " where archived_flag = [archived] "

    ' Find the highest X-component of the version number
    sBuf = sBuf & " AND cint( mid( version_no, 1, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS d " & _
        " WHERE a.step_id = d.step_id ) "

    ' Find the highest Y-component of the version number for the highest
X-component
    sBuf = sBuf & " AND cint( mid( version_no, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) + 1 ) ) = " & _
        " ( select max( cint( mid( version_no, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) + 1 ) ) ) " & _
        " from att_steps AS b " & _
        " Where a.step_id = b.step_id " & _
        " AND cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS c " & _
        " WHERE a.step_id = c.step_id ) ) "

    ' Create a temporary Querydef object
    Set qyTemp = dbFile.CreateQueryDef(gstrEmptyString, sBuf)
    qyTemp.Parameters("archived").Value = False

    qyTemp.Execute dbFailOnError
    qyTemp.Close

    Exit Sub

UpdateContinuationCriteriaErr:
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errModifyStepFailed, mstrModuleName, _
            LoadResString(errModifyStepFailed)

End Sub

Private Sub UpdateDbDtls(dbFile As Database, sNewVersion As String)

    Dim sSql As String
    Dim cTemp As New cStringSM

    On Error GoTo UpdateDbDtlsErr

    sSql = "update db_details " & _
            " set db_version = " &
cTemp.MakeStringFieldValid(sNewVersion)

    dbFile.Execute sSql, dbFailOnError

    Exit Sub

UpdateDbDtlsErr:
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
            LoadResString(errUpgradeFailed)

End Sub

Private Sub Upgrade10to21(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sSql As String

    On Error GoTo Upgrade10to21Err

    Call UpdateDbDtls(dbFile, sVersion)

    Call UpdateContinuationCriteria(dbFile)

    Exit Sub

Upgrade10to21Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
            LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade21to23(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sBuf As String
    Dim cTempStr As New cStringSM

    On Error GoTo Upgrade21to23Err

    ' Add a parameter type field and a description field to the
parameter table
    sBuf = "alter table workspace_parameters " & _
            " add column description TEXT(255) "
```

```
    dbFile.Execute sBuf, dbFailOnError

    sBuf = "alter table workspace_parameters " & _
           " add column parameter_type INTEGER "
    dbFile.Execute sBuf, dbFailOnError

    ' Initialize the parameter type on all parameters to indicate
generic parameters
    sBuf = "update workspace_parameters " & _
           " set parameter_type = " & CStr(gintParameterGeneric)
    dbFile.Execute sBuf, dbFailOnError

    sBuf = "Release 2.3 onwards, connection string parameters will be "
& _
           "displayed in a separate node. After this upgrade, all
connection " & _
           "string parameters will appear under the Globals/Connection
Strings " & _
           "node in the workspace. "
    Call MsgBox(sBuf, vbOKOnly + vbApplicationModal, "Upgrade database")

    ' Update the parameter type on all parameters that look like db
connection strings
    sBuf = "update workspace_parameters " & _
           " set parameter_type = " & CStr(gintParameterConnect) & _
           " where UCase(parameter_value) like '*DRIVER*' " & _
           " or UCase(parameter_value) like '*DSN*'"
    dbFile.Execute sBuf, dbFailOnError

    ' Add an elapsed time field to the run_step_details table - this
field is
    ' needed to store the elapsed time in milliseconds.
    sBuf = "alter table run_step_details " & _
           " add column elapsed_time LONG "
    dbFile.Execute sBuf, dbFailOnError

    ' The failure_details field has some data for the case when an ODBC
failure
    ' threshold was specified. Since that's no longer relevant, update
the failure_details
    ' field for records with failure_criteria = gintFailureODBC to empty.
    ' failure_criteria = gintFailureODBC = 1
    sBuf = "update att_steps " & _
           " set failure_details = " &
cTempStr.MakeStringFieldValid(gstrEmptyString) & _
           " where failure_criteria = '1'"
    dbFile.Execute sBuf, dbFailOnError

    Call UpdateDbDtls(dbFile, sVersion)

    UpgradeWsp.CommitTrans

    On Error GoTo DropColumnErr

    UpgradeWsp.BeginTrans

    ' This ddl cannot be in the same transaction as the failure_details
update
    ' But we can do this in a separate transaction since we do not
expect this
    ' statement to fail - AND, it doesn't matter if this transaction
fails
    ' Drop the failure_criteria column from the att_steps table
    sBuf = "alter table att_steps " & _
           " drop column failure_criteria "
    dbFile.Execute sBuf, dbFailOnError

    Exit Sub

DropColumnErr:
    Call LogErrors(Errors)
    ShowError errDeleteColumnFailed
    Exit Sub

Upgrade21to23Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
           LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade23to24(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sBuf As String
    Dim lId As Long
    Dim rTemp As DAO.Recordset
    Dim cTempStr As New cStringSM

    On Error GoTo Upgrade23to24Err

    ' Add a new table for connection properties
    sBuf = CreateConnectionsTableScript()
    ' TODO: Not sure of column sizes for row count, tsql_batch_separator
and server_language
    dbFile.Execute sBuf, dbFailOnError

    ' Move all connection parameters from the parameter table to the
connections tables
    ' Insert default values for the newly added connection properties
    sBuf = "select * from workspace_parameters " & _
           "where parameter_type = " & CStr(gintParameterConnect)
    Set rTemp = dbFile.OpenRecordset(sBuf, dbOpenSnapshot)
    lId = 1
    If rTemp.RecordCount <> 0 Then
        rTemp.MoveFirst

        While Not rTemp.EOF
            sBuf = "insert into workspace_connections " & _
             "( workspace_id, connection_id, " & _
             "connection_name, connection_value, " & _
             "description, no_count_display, " & _
```

```
            "no_execute, parse_query_only, " & _
            "ANSI_quoted_identifiers, ANSI_nulls, " & _
            "show_query_plan, show_stats_time, " & _
            "show_stats_io, parse_odbc_msg_prefixes, " & _
            "row_count, tsql_batch_separator, " & _
            "query_time_out, server_language, " & _
            "character_translation, regional_settings ) " & _
           " values ( " & _
           Str(rTemp!workspace_id) & ", " & Str(lId) & ", " & _
           cTempStr.MakeStringFieldValid("" & rTemp!parameter_name) & ",
" & _
           cTempStr.MakeStringFieldValid("" & rTemp!parameter_value) &
", " & _
           cTempStr.MakeStringFieldValid("" & rTemp!Description) & ", "
& _
           Str(DEF_NO_COUNT_DISPLAY) & ", " & _
           Str(DEF_NO_EXECUTE) & ", " & Str(DEF_PARSE_QUERY_ONLY) & ",
" & _
           Str(DEF_ANSI_QUOTED_IDENTIFIERS) & ", " &
Str(DEF_ANSI_NULLS) & ", " & _
           Str(DEF_SHOW_QUERY_PLAN) & ", " & Str(DEF_SHOW_STATS_TIME) &
", " & _
           Str(DEF_SHOW_STATS_IO) & ", " &
Str(DEF_PARSE_ODBC_MSG_PREFIXES) & ", " & _
           Str(DEF_ROW_COUNT) & ", " &
cTempStr.MakeStringFieldValid(DEF_TSQL_BATCH_SEPARATOR) & ", " & _
           Str(DEF_QUERY_TIME_OUT) & ", " &
cTempStr.MakeStringFieldValid(DEF_SERVER_LANGUAGE) & ", " & _
           Str(DEF_CHARACTER_TRANSLATION) & ", " &
Str(DEF_REGIONAL_SETTINGS) & _
           " ) "
           dbFile.Execute sBuf, dbFailOnError

           lId = lId + 1
           rTemp.MoveNext
       Wend
    End If
    rTemp.Close

    ' Add an identifier column for the connection_id field
    sBuf = "alter table att_identifiers " & _
           " add column connection_id long "
    dbFile.Execute sBuf, dbFailOnError

    ' Initialize the value of the connection identifier, initialized
above
    sBuf = "update att_identifiers " & _
           " set connection_id = " & Str(lId)
    dbFile.Execute sBuf, dbFailOnError

    ' Delete all connection strings from the parameter table
    sBuf = "delete from workspace_parameters " & _
            "where parameter_type = " & CStr(gintParameterConnect)
    dbFile.Execute sBuf, dbFailOnError

    ' Create the built-in parameter, default directory, for each
workspace in the db
    Call InsertBuiltInParameter(dbFile, PARAM_DEFAULT_DIR,
gstrEmptyString, PARAM_DEFAULT_DIR_DESC)

    Call UpdateDbDtls(dbFile, sVersion)

    Exit Sub

Upgrade23to24Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
            LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade243to25(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sBuf As String
    Dim qy As DAO.QueryDef
    Dim rTemp As DAO.Recordset
    Dim lId As Long
    Dim cTempStr As New cStringSM

    On Error GoTo Upgrade243to25Err

    sBuf = "Release " & gsVersion25 & " onwards, new 'Connections' must
be created for all " & _
           "connection strings. " & vbCrLf & vbCrLf & _
           "Connections will appear under the Globals/Connections " & _
           "node in the workspace. " & vbCrLf & _
           "A list of all 'Connections' (instead of 'Connection
Strings') " & _
           "in the workspace will be displayed in the 'Connections'
field for " & _
           "ODBC steps on the Step definition screen. " & vbCrLf &
vbCrLf & _
           "Each Connection can be marked as static or dynamic. " &
vbCrLf & _
           "Dynamic connections will be created when a step starts
execution and " & _
           "closed once the step completes. " & vbCrLf & _
           "Static connections will be kept open till the run
completes." & vbCrLf & vbCrLf & _
           "Currently dynamic 'Connections' have been created for all
existing 'Connection Strings' " & _
           "with the suffix " & CONNECTION_STRINGS_TO_NAME_SUFFIX
    Call MsgBox(sBuf, vbOKOnly + vbApplicationModal, "Upgrade database")

    ' Add a new table for the connection name entity
    ' This table has been added in order to satisfy the TPC-H
requirement that
    ' all the queries in a stream need to be executed on a single
connection.
    sBuf = CreateConnectionDtlsTableScript()
```

```
    dbFile.Execute sBuf, dbFailOnError

    ' Add an identifier column for the connection_name_id field
    sBuf = "alter table att_identifiers " & _
            " add column " & FLD_ID_CONN_NAME & " long "
    dbFile.Execute sBuf, dbFailOnError

    Call UpdateDbDtls(dbFile, sVersion)

    ' insert connection_dtl records for each of the connection strings
    sBuf = "select * from " & TBL_CONNECTION_STRINGS
    Set rTemp = dbFile.OpenRecordset(sBuf, dbOpenSnapshot)

    sBuf = "insert into " & TBL_CONNECTION_DTLS & _
             "( " & FLD_ID_WORKSPACE & _
             ", " & FLD_ID_CONN_NAME & _
             ", " & FLD_CONN_DTL_CONNECTION_NAME & _
             ", " & FLD_CONN_DTL_CONNECTION_STRING & _
             ", " & FLD_CONN_DTL_CONNECTION_TYPE & " ) " & _
            " values ( [w_id], [c_id], [c_name], [c_str], [c_type] ) "
    Set qy = dbFile.CreateQueryDef("", sBuf)

    lId = glMinId
    If rTemp.RecordCount <> 0 Then
        rTemp.MoveFirst

        While Not rTemp.EOF
            qy.Parameters("w_id").Value = rTemp.Fields(FLD_ID_WORKSPACE)
            qy.Parameters("c_id").Value = lId
            qy.Parameters("c_name").Value =
rTemp.Fields(FLD_CONN_STR_CONNECTION_NAME) &
CONNECTION_STRINGS_TO_NAME_SUFFIX
            qy.Parameters("c_str").Value =
rTemp.Fields(FLD_CONN_STR_CONNECTION_NAME)
            qy.Parameters("c_type").Value = ConnTypeDynamic

            qy.Execute dbFailOnError

            lId = lId + 1
            rTemp.MoveNext
        Wend
    End If
    qy.Close
    rTemp.Close

    ' Initialize the value of the connection_name_id
    sBuf = "update att_identifiers " & _
            " set " & FLD_ID_CONN_NAME & " = " & Str(lId)
    dbFile.Execute sBuf, dbFailOnError

    ' Update the start_directory field in att_steps to point to the
newly
    ' created connections
    Call ReadStepsInWorkspace(rTemp, qy, glInvalidId, dbLoad:=dbFile, _
            bSelectArchivedRecords:=False)

    sBuf = "update " & TBL_STEPS & _
        " set " & FLD_STEPS_EXEC_DTL & " = [c_name] " & _
        " where " & FLD_ID_STEP & " = [s_id] " & _
        " and " & FLD_STEPS_VERSION_NO & " = [ver_no] "
    Set qy = dbFile.CreateQueryDef("", sBuf)

    If rTemp.RecordCount <> 0 Then
        rTemp.MoveFirst

        While Not rTemp.EOF
            If rTemp.Fields(FLD_STEPS_EXEC_MECHANISM).Value =
gintExecuteODBC Then
                If Not (StringEmpty("" &
rTemp.Fields(FLD_STEPS_EXEC_DTL))) Then
                    sBuf = rTemp.Fields(FLD_STEPS_EXEC_DTL)
                    ' Strip the enclosing "%" characters
                    sBuf = Mid(sBuf, 2, Len(sBuf) - 2) &
CONNECTION_STRINGS_TO_NAME_SUFFIX

                    qy.Parameters("c_name").Value = sBuf
                    qy.Parameters("s_id").Value =
rTemp.Fields(FLD_ID_STEP)
                    qy.Parameters("ver_no").Value =
rTemp.Fields(FLD_STEPS_VERSION_NO)

                    qy.Execute dbFailOnError
                End If
            End If
            rTemp.MoveNext
        Wend
    End If

    qy.Close
    rTemp.Close

    Exit Sub

Upgrade243to25Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
            LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade25to251(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    On Error GoTo Upgrade25to251Err

    ' Create the built-in parameters, run_id and output_dir, for each
workspace in the db
    Call InsertBuiltInParameter(dbFile, PARAM_RUN_ID, gstrEmptyString,
PARAM_RUN_ID_DESC)
    Call InsertBuiltInParameter(dbFile, PARAM_OUTPUT_DIR,
gstrEmptyString, PARAM_OUTPUT_DIR_DESC)
```

```
    Call UpdateDbDtls(dbFile, sVersion)

    Exit Sub

Upgrade25to251Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
            LoadResString(errUpgradeFailed)

End Sub

Private Sub Upgrade242to243(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sBuf As String
    Dim cTempStr As New cStringSM
    Dim iResponse As Integer

    On Error GoTo DeleteHistoryErr

    Call DeleteRunHistory(dbFile)

    On Error GoTo Upgrade242to243Err

    UpgradeWsp.CommitTrans

    UpgradeWsp.BeginTrans

    ' Add a parameter type field and a description field to the
parameter table
    sBuf = "alter table run_step_details " & _
            " add column parent_instance_id LONG "

    dbFile.Execute sBuf, dbFailOnError

    sBuf = "alter table run_step_details " & _
            " add column iterator_value TEXT(255) "

    dbFile.Execute sBuf, dbFailOnError

    Call AlterFieldType(dbFile, TBL_RUN_STEP_DTLS, "start_time",
DATA_TYPE_CURRENCY)
    Call AlterFieldType(dbFile, TBL_RUN_STEP_DTLS, "end_time",
DATA_TYPE_CURRENCY)
    Call AlterFieldType(dbFile, TBL_RUN_STEP_HDR, "start_time",
DATA_TYPE_CURRENCY)
    Call AlterFieldType(dbFile, TBL_RUN_STEP_HDR, "end_time",
DATA_TYPE_CURRENCY)

    Call UpdateDbDtls(dbFile, sVersion)

    Exit Sub

DeleteHistoryErr:
    ' This is not a critical error - continue with upgrade
    Call LogErrors(Errors)
    Resume Next

Upgrade242to243Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
            LoadResString(errUpgradeFailed)

End Sub
'****************************************************************
' The AlterFieldType Sub procedure requires three string
' parameters. The first string specifies the name of the table
' containing the field to be changed. The second string specifies
' the name of the field to be changed. The third string specifies
' the new data type for the field.
'****************************************************************

Private Sub AlterFieldType(dbFile As Database, TblName As String,
FieldName As String, _
                    NewDataType As String)
    Dim qdf As DAO.QueryDef
    Dim sSql As String

    ' Add a temporary field to the table.
    sSql = "ALTER TABLE [" & TblName & _
            "] ADD COLUMN AlterTempField " & NewDataType
    Set qdf = dbFile.CreateQueryDef("", sSql)
    qdf.Execute

    ' Copy the data from old field into the new field.
    qdf.SQL = "UPDATE DISTINCTROW [" & TblName & "] SET AlterTempField =
[" & FieldName & "]"
    qdf.Execute

    ' Delete the old field.
    qdf.SQL = "ALTER TABLE [" & TblName & "] DROP COLUMN [" & FieldName
& "]"
    qdf.Execute

    ' Rename the temporary field to the old field's name.
    dbFile.TableDefs("[" & TblName & "]").Fields("AlterTempField").Name
= FieldName
    dbFile.TableDefs.Refresh

    ' Clean up.
End Sub
Private Sub Upgrade01to21(UpgradeWsp As DAO.Workspace, dbFile As
DAO.Database, sVersion As String)
    Dim sSql As String

    On Error GoTo Upgrade01to21Err

    sSql = "Create table db_details (" & _
             "db_version                Text(50) " & _
    ");"
```

```
    dbFile.Execute sSql, dbFailOnError

    sSql = "insert into db_details " & _
            "( db_version ) values ( '" & sVersion & "' ) "

    dbFile.Execute sSql, dbFailOnError

    Call UpdateContinuationCriteria(dbFile)

    Exit Sub

Upgrade01to21Err:
    Call LogErrors(Errors)
    UpgradeWsp.Rollback
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
            LoadResString(errUpgradeFailed)

End Sub
Private Function UpgradeDb(UpgradeWsp As DAO.Workspace, dbFile As
Database, _
        sVerTo As String, sVerFrom As String) As Boolean

    Dim sMsg As String

    On Error GoTo UpgradeDbErr

    UpgradeDb = False
    If Not ValidUpgrade(sVerTo, sVerFrom) Then Exit Function

    If NoDbChanges(sVerTo, sVerFrom) Then
        UpgradeDb = True
        Exit Function
    End If

    sMsg = "The database needs to be upgraded from Version " & sVerFrom
& _
            " to Version " & sVerTo & "." & vbCrLf & _
            "Proceed?"
    If Not Confirm(Buttons:=vbYesNo, strMessage:=sMsg,
strTitle:="Upgrade database") Then
        Exit Function
    End If

    UpgradeWsp.BeginTrans

    Select Case sVerFrom
        Case gsVersion25
            Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

        Case gsVersion243
            Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
            Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

        Case gsVersion24, gsVersion241, gsVersion242
            sMsg = "After this upgrade, the run history for previous
runs will no longer be available. " & _
                    "Continue?"
            If Not Confirm(Buttons:=vbYesNo, strMessage:=sMsg,
strTitle:="Upgrade database") Then
                UpgradeWsp.CommitTrans
                Exit Function
            End If
            Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion243)
            Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
            Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

        Case gsVersion23
            Call Upgrade23to24(UpgradeWsp, dbFile, gsVersion24)
            Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion242)
            Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
            Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

        Case gsVersion21
            Call Upgrade21to23(UpgradeWsp, dbFile, gsVersion23)
            Call Upgrade23to24(UpgradeWsp, dbFile, gsVersion24)
            Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion242)
            Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
            Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

        Case gsVersion10
            Call Upgrade10to21(UpgradeWsp, dbFile, gsVersion21)
            Call Upgrade21to23(UpgradeWsp, dbFile, gsVersion23)
            Call Upgrade23to24(UpgradeWsp, dbFile, gsVersion24)
            Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion242)
            Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
            Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

        Case gsVersion01
            Call Upgrade01to21(UpgradeWsp, dbFile, gsVersion21)
            Call Upgrade21to23(UpgradeWsp, dbFile, gsVersion23)
            Call Upgrade23to24(UpgradeWsp, dbFile, gsVersion24)
            Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion242)
            Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
            Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

    End Select

    UpgradeWsp.CommitTrans

    UpgradeDb = True
    Exit Function

UpgradeDbErr:
    Call LogErrors(Errors)
    ShowError errUpgradeFailed

End Function
Private Function DBVersion(TestDb As Database) As String
    ' Retrieves the database version
    Dim rVersion As Recordset
```

```
    On Error GoTo DBVersionErr

    Set rVersion = TestDb.OpenRecordset("Select db_version from
db_details ", _
            dbOpenForwardOnly)

    BugAssert rVersion.RecordCount <> 0
    DBVersion = rVersion!db_version

    rVersion.Close
    Exit Function

DBVersionErr:
    If Err.Number = merrDaoTableMissing Then
        DBVersion = gsVersion01
    Else
        LogErrors Errors
        Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
                LoadResString(errUpgradeFailed)
    End If

End Function

Private Function ValidUpgrade(sVerTo As String, sVerFrom As String) As
Boolean

    If sVerTo = gsVersion And sVerFrom = gsVersion251 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion25 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion243 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion242 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion241 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion24 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion23 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion21 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion10 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion01 Then
        ValidUpgrade = True
    Else
        ValidUpgrade = False
    End If

End Function
```

DebugSM.bas

```
Attribute VB_Name = "DebugSM"
'  FILE:       DebugSM.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains all the functions that carry out error/debug
'              processing for the project.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
' Most of the functions in this module that manipulate the
' error object do not have an On Error GoTo statement - this
' is because it will clear the passed in error object - let
' the calling functions handle the errors raised by this
' module, if any
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "DebugSM."

Private mcLogFile As cFileSM
Private mcErrorFile As cFileSM

Private Const FORMAT_MESSAGE_FROM_SYSTEM = &H1000
Private Const FORMAT_MESSAGE_IGNORE_INSERTS = &H200
Private Const pNull = 0

Declare Function FormatMessage Lib "kernel32" Alias "FormatMessageA"
(ByVal dwFlags As Long, lpSource As Any, ByVal dwMessageId As Long,
ByVal dwLanguageId As Long, ByVal lpbuffer As String, ByVal nSize As
Long, Arguments As Long) As Long
Public Function Confirm(Optional lngMessageCode As conConfirmMsgCodes, _
        Optional lngTitleCode As conConfirmMsgTitleCodes, _
        Optional TitleParameter As String, _
        Optional ByVal Buttons As Integer = -1, _
        Optional strMessage As String = gstrEmptyString, _
        Optional strTitle As String = gstrEmptyString) _
        As Boolean
    ' Displays a confirmation message corresponding to the
    ' passed in message code. Returns True if the user says
    ' Ok and False otherwise

    Dim intResponse As Integer
    Dim intButtonStyle As Integer

    On Error GoTo ConfirmErr

    Confirm = False

    ' If the buttons style hasn't been specified, set the
    ' default style to display OK and Cancel buttons
    If Buttons = -1 Then
        intButtonStyle = vbOKCancel
    Else
        intButtonStyle = Buttons
```

```
    End If

    ' Find the message string for the passed in code
    If StringEmpty(strMessage) Then
        strMessage = Trim$(LoadResString(lngMessageCode))
    End If

    If StringEmpty(strTitle) Then
        strTitle = Trim$(LoadResString(lngTitleCode))
    End If

    If Not StringEmpty(TitleParameter) Then
        strTitle = strTitle & Chr$(vbKeySpace) & _
                gstrSQ & TitleParameter & gstrSQ
    End If

    ' Display the confirmation message with the Cancel button
    ' set to the default - assume that we are confirming
    ' potentially dangerous operations!
    intResponse = MsgBox(strMessage, _
            intButtonStyle + vbQuestion + vbApplicationModal, _
            strTitle)

    ' Translate the user response into a True/False return code
    If intButtonStyle = vbOKCancel Then
        If intResponse = vbOK Then
            Confirm = True
        Else
            Confirm = False
        End If
    Else
        If intResponse = vbYes Then
            Confirm = True
        Else
            Confirm = False
        End If
    End If

    Exit Function

ConfirmErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    gstrSource = mstrModuleName & "Confirm"
    Err.Raise vbObjectError + errConfirmFailed, _
            gstrSource, _
            LoadResString(errConfirmFailed)

End Function
Public Sub LogSystemError()
    Dim eErrCode As Long

    eErrCode = GetLastError()
    If eErrCode <> 0 Then
        WriteToFile "System Error: " & eErrCode & vbCrLf &
ApiError(eErrCode), _
                blnError:=True
    End If

End Sub
Public Function ApiError(ByVal e As Long) As String

    Dim s As String
    Dim c As Long

    s = String(256, 0)
    c = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM Or _
            FORMAT_MESSAGE_IGNORE_INSERTS, _
            pNull, e, 0&, s, Len(s), ByVal pNull)
    If c Then ApiError = e & ": " & Left$(s, c)

End Function

' Output flags determine output destination of BugAsserts and messages
#Const afLogfile = 1
#Const afMsgBox = 2
#Const afDebugWin = 4
#Const afAppLog = 8

' Display appropriate error message, and then stop
' program.  These errors should NOT be possible in
' shipping product.
Sub BugAssert(ByVal fExpression As Boolean, _
            Optional sExpression As String)
#If afDebug Then
    If fExpression Then Exit Sub
    BugMessage "BugAssert failed: " & sExpression
    Stop
#End If
End Sub

Sub BugMessage(sMsg As String)

#If afDebug And afLogfile Then
    ' Since we are writing log messages, the error flag is turned off
    Call WriteToFile(sMsg, False)
#End If
#If afDebug And afMsgBox Then
    MsgBox sMsg
#End If
#If afDebug And afDebugWin Then
    Debug.Print sMsg
#End If
#If afDebug And afAppLog Then
    App.LogEvent sMsg
#End If

End Sub
Public Function ProjectLogFile() As String
```

```
    ProjectLogFile = mcLogFile.FileName

End Function
Public Function ProjectErrorFile() As String

    ProjectErrorFile = mcErrorFile.FileName

End Function

Private Sub WriteToFile(sMsg As String, Optional ByVal blnError As
Boolean)

    ' Calls procedures to write the passed in message to the log -
    ' The blnError flag is used to indicate that the message
    ' should be logged to the error file - by default the log
    ' file is used

    Dim mcFileObj As cFileSM
    Dim strFileName As String
    Dim strFileHdr As String

    On Error GoTo WriteToFileErr

    If blnError Then
        If mcErrorFile Is Nothing Then
            Set mcErrorFile = New cFileSM
        End If
        Set mcFileObj = mcErrorFile
    Else
        If mcLogFile Is Nothing Then
            Set mcLogFile = New cFileSM
        End If
        Set mcFileObj = mcLogFile
    End If

    If StringEmpty(mcFileObj.FileName) Then
        If blnError Then
            strFileName = gstrProjectPath & "\" & App.EXEName & ".ERR"
            strFileHdr = "Stepmaster Errors"
        Else
            strFileName = gstrProjectPath & "\" & App.EXEName & ".DBG"
            strFileHdr = "Stepmaster Log"
        End If

        mcFileObj.FileName = strFileName
        mcFileObj.WriteLine strFileHdr
        mcFileObj.WriteLine "Log start time : " & Now
    End If

    mcFileObj.WriteLine sMsg

    Exit Sub

WriteToFileErr:
    ' Display the error code raised by Visual Basic
    Call DisplayErrors(Errors)
    ' An error message would've been displayed by the called
    ' procedures

End Sub
Public Sub WriteMessage(sMsg As String)

    Call WriteToFile(sMsg, True)

End Sub

Sub BugTerm()
#If afDebug And afLogfile Then
    ' Close log file
    mcLogFile.CloseFile
#End If
End Sub

Public Sub ShowError(ByVal ErrorCode As errErrorConstants, _
        Optional ByVal ErrorSource As String = gstrEmptyString, _
        Optional ByVal OptArgs As String = gstrEmptyString, _
        Optional ByVal DoWriteError As Boolean = True)

    If DoWriteError Then
        ' Call a procedure to write the error to a log file
        Call WriteError(ErrorCode, ErrorSource, OptArgs)
    End If

    ' Re-initialize the values of the Error object before
    ' displaying the error to the user
    Call InitErrObject(ErrorCode, ErrorSource, OptArgs)

    Call DisplayErrors(Errors)

    Err.Clear

End Sub
Public Sub WriteError(ByVal ErrorCode As errErrorConstants, _
        Optional ByVal ErrorSource As String = gstrEmptyString, _
        Optional ByVal OptArgs As String = gstrEmptyString)

    ' Initialize the values of the Error object before
    ' calling the log function
    Call InitErrObject(ErrorCode, ErrorSource, OptArgs)

    Call LogErrors(Errors)

    Err.Clear

End Sub
Private Sub InitErrObject(ByVal ErrorCode As errErrorConstants, _
        Optional ByVal ErrorSource As String = gstrEmptyString, _
        Optional ByVal OptArgs As String = gstrEmptyString)

    Dim lngError As Long
```

```
    lngError = IIf(ErrorCode > vbObjectError And ErrorCode <
vbObjectError + 65535, _
            ErrorCode - vbObjectError, ErrorCode)
    Err.Number = lngError + vbObjectError
    Err.Description = LoadResString(lngError) & OptArgs
    Err.Source = App.EXEName & ErrorSource

End Sub
Public Sub ShowMessage(ByVal MessageCode As errErrorConstants, _
        Optional ByVal OptArgs As String)

    Dim strMessage As String

    On Error GoTo ShowMessageErr

    strMessage = LoadResString(MessageCode) & OptArgs

    ' Write the error to a log file
    BugMessage strMessage

    MsgBox strMessage, vbOKOnly

    Exit Sub

ShowMessageErr:
    ' Log the error and exit
    Call DisplayErrors(Errors)

End Sub
Public Sub ShowMessageStr(sMessage As String)

    ' Write the error to a log file
    BugMessage sMessage

    MsgBox sMessage, vbOKOnly

End Sub

Public Sub DisplayErrors(myErrCollection As Errors)
    Dim strError As String
    Dim errLoop As Error
    Dim errCode As Long

    ' Enumerate Errors collection and display properties of
    ' each Error object.
    If Err.Number <> 0 Then
        If Err.Number > vbObjectError And Err.Number < (vbObjectError +
65536) Then
            errCode = Err.Number - vbObjectError
        Else
            errCode = Err.Number
        End If
        strError = "Error # " & Str(errCode) & " was generated by " _
            & Err.Source & Chr(13) & Err.Description
        MsgBox strError, , "Error", Err.HelpFile, Err.HelpContext
    Else
        For Each errLoop In myErrCollection
            With errLoop
                If Err.Number > vbObjectError And Err.Number <
(vbObjectError + 65536) Then
                    errCode = .Number - vbObjectError
                Else
                    errCode = .Number
                End If
                strError = "Error #" & errCode & vbCrLf
                strError = strError & "    " & .Description & vbCrLf
                strError = strError & _
                    "    (Source: " & .Source & ")" & vbCrLf
'                strError = strError & _
'                    "Press F1 to see topic " & .HelpContext & vbCrLf
'                strError = strError & _
'                    "    in the file " & .HelpFile & "."
            End With

            MsgBox strError
        Next
    End If

End Sub
Public Sub LogErrors(myErrCollection As Errors)
    Dim cColErrors As cVectorStr
    Dim strError As String
    Dim errLoop As Error
    Dim errCode As Long
    Dim lngIndex As Long

    Set cColErrors = New cVectorStr

    ' Enumerate Errors collection and display properties of
    ' each Error object.
    If Err.Number <> 0 Then
        If Err.Number > vbObjectError And Err.Number < (vbObjectError +
65536) Then
            errCode = Err.Number - vbObjectError
        Else
            errCode = Err.Number
        End If
        strError = "Error # " & Str(errCode) & " was generated by " _
            & Err.Source & vbCrLf & Err.Description

        cColErrors.Add strError
    End If

    ' Log all database errors, if any
    For Each errLoop In myErrCollection
        With errLoop
            If Err.Number > vbObjectError And Err.Number <
(vbObjectError + 65536) Then
                errCode = .Number - vbObjectError
            Else
                errCode = .Number
            End If
```

```
            strError = "Error #" & errCode & vbCrLf
            strError = strError & "    " & .Description & vbCrLf
            strError = strError & _
                "    (Source: " & .Source & ")" & vbCrLf
        End With

        cColErrors.Add strError
    Next

    ' We can have a error handler now that we have stored all
    ' errors away safely! - having an error handler before
    ' enumerating all the errors would have cleared the error
    ' collection
    On Error GoTo LogErrorsErr
    gstrSource = mstrModuleName & "LogErrors"

    For lngIndex = 0 To cColErrors.Count - 1
        strError = cColErrors(lngIndex)
        Debug.Print strError
        Call WriteToFile(strError, True)
    Next lngIndex

    Set cColErrors = Nothing

    Exit Sub

LogErrorsErr:
    ' Display the error code raised by Visual Basic
    DisplayErrors Errors
    On Error GoTo 0
    ShowError errUnableToWriteError, DoWriteError:=False

End Sub
```

FileCommon.bas

```
Attribute VB_Name = "FileCommon"
'  FILE:       FileCommon.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module contains common functionality to display
'              the File Open dialog.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "FileCommon."

Private Enum EOpenFile
    OFN_OVERWRITEPROMPT = &H2
    OFN_HIDEREADONLY = &H4
    OFN_FILEMUSTEXIST = &H1000
    OFN_EXPLORER = &H80000
End Enum

' The locations for the different output files are presented to
' the user in a list box. These constants are used while loading the
' data and while reading the data from the list box.
' These constants also represent the different file types that are
' displayed to the user in File Open dialogs
Public Enum gFileTypes
    gintOutputFile = 0
'    gintLogFile = 1
    gintErrorFile
    gintStepTextFile
    gintOutputCompareFile
    gintDBFile
    gintDBFileNew
    gintImportFile
    gintExportFile
End Enum

Public Const gsSqlFileSuffix = ".sql"
Public Const gsCmdFileSuffix = ".cmd"

Public Const gsOutputFileSuffix = ".out"
Public Const gstrLogFileSuffix = ".log"
Public Const gsErrorFileSuffix = ".err"
Public Function BrowseDBFile() As String
    ' Prompts the user for a database file with the workspace
information
    ' Call CallFileDialog to display the open file dialog
    BrowseDBFile = CallFileDialog(gintDBFile)

End Function
Public Function CallFileDialog(intFileType As Integer, _
        Optional ByVal strDefaultFile As String = gstrEmptyString) As
String
    ' This function initializes the values of the filter property,
    ' the dialog title and flags for the File Open dialog depending
    ' on the FileType passed in
    ' It then calls ShowFileOpenDialog to set these properties and
    ' display the File Open dialog to the user

    ' All the properties used by the File Open dialog are defined
    ' as constants in this function and passed to ShowFileOpenDialog
    ' as parameters. So if any of the dialog properties need to be
    ' modified, these constants are what need to be changed
    Const s_DLG_TITLE_OPEN = "Open"
    Const s_DLG_TITLE_NEW = "New"
    Const s_DLG_TITLE_IMPORT = "Import From"
    Const s_DLG_TITLE_EXPORT = "Export To"

    Const mlng_FILE_STEP_TEXT_FLAGS = OFN_EXPLORER Or OFN_FILEMUSTEXIST
Or OFN_HIDEREADONLY
    Const mlng_FILE_OUTPUT_COMPARE_FLAGS = mlng_FILE_STEP_TEXT_FLAGS
    Const mlng_FILE_DB_FLAGS = mlng_FILE_STEP_TEXT_FLAGS
```

```
    Const mlng_FILE_OUTPUT_FLAGS = OFN_EXPLORER Or OFN_HIDEREADONLY Or
OFN_OVERWRITEPROMPT
    Const mlng_FILE_LOG_FLAGS = mlng_FILE_OUTPUT_FLAGS
    Const mlng_FILE_ERROR_FLAGS = mlng_FILE_OUTPUT_FLAGS
    Const mlng_FILE_DB_NEW_FLAGS = mlng_FILE_OUTPUT_FLAGS

    Const mstr_FILE_ALL_FILTER = "|All Files (*.*)|*.*"
    Const mstr_FILE_STEP_TEXT_FILTER = "Query Files (*" &
gsSqlFileSuffix & _
            ")|*" & gsSqlFileSuffix & "|Command Script Files (*" &
gsCmdFileSuffix & _
            ")|*" & gsCmdFileSuffix
    Const mstr_FILE_OUTPUT_COMPARE_FILTER = "Text Files (*.txt)|*.txt"
    Const mstr_FILE_OUTPUT_FILTER = "Output Files (*.out)|*.out"
    Const mstr_FILE_LOG_FILTER = "Log Files (*.log)|*.log"
    Const mstr_FILE_ERROR_FILTER = "Error Files (*.err)|*.err"
    Const mstr_FILE_DB_FILTER = "Stepmaster Workspace Files (*" &
gsDefDBFileExt & ")|*" & gsDefDBFileExt

    Dim strFileName As String

    On Error GoTo CallFileDialogErr

    Select Case intFileType
        Case gintStepTextFile
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_STEP_TEXT_FILTER & mstr_FILE_ALL_FILTER, _
                s_DLG_TITLE_OPEN, _
                mlng_FILE_STEP_TEXT_FLAGS, _
                strDefaultFile)

        Case gintOutputCompareFile
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_OUTPUT_COMPARE_FILTER & mstr_FILE_ALL_FILTER,
_
                s_DLG_TITLE_OPEN, _
                mlng_FILE_OUTPUT_COMPARE_FLAGS, _
                strDefaultFile)

        Case gintOutputFile
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_OUTPUT_FILTER & mstr_FILE_ALL_FILTER, _
                s_DLG_TITLE_OPEN, _
                mlng_FILE_OUTPUT_FLAGS, _
                strDefaultFile)

'        Case gintLogFile
'            strFileName = ShowFileOpenDialog( _
'                mstr_FILE_LOG_FILTER & mstr_FILE_ALL_FILTER, _
'                s_DLG_TITLE_OPEN, _
'                mlng_FILE_LOG_FLAGS, _
'                strDefaultFile)
'
        Case gintErrorFile
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_ERROR_FILTER & mstr_FILE_ALL_FILTER, _
                s_DLG_TITLE_OPEN, _
                mlng_FILE_ERROR_FLAGS, _
                strDefaultFile)

        Case gintDBFile
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_DB_FILTER & mstr_FILE_ALL_FILTER, _
                s_DLG_TITLE_OPEN, _
                mlng_FILE_DB_FLAGS, _
                strDefaultFile)

        Case gintDBFileNew
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_DB_FILTER & mstr_FILE_ALL_FILTER, _
                s_DLG_TITLE_NEW, _
                mlng_FILE_DB_NEW_FLAGS, _
                strDefaultFile)

        Case gintImportFile
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_DB_FILTER & mstr_FILE_ALL_FILTER, _
                s_DLG_TITLE_IMPORT, _
                mlng_FILE_DB_FLAGS, _
                strDefaultFile)

        Case gintExportFile
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_DB_FILTER & mstr_FILE_ALL_FILTER, _
                s_DLG_TITLE_EXPORT, _
                mlng_FILE_DB_FLAGS, _
                strDefaultFile)

        Case Else
            BugAssert True, "Incorrect file type passed in."
            ' Default processing will be for the output file
            strFileName = ShowFileOpenDialog( _
                mstr_FILE_OUTPUT_FILTER & mstr_FILE_ALL_FILTER, _
                s_DLG_TITLE_OPEN, _
                mlng_FILE_OUTPUT_FLAGS, _
                strDefaultFile)

    End Select
    CallFileDialog = strFileName

    Exit Function

CallFileDialogErr:
    CallFileDialog = gstrEmptyString
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "CallFileDialog"
    Call ShowError(errBrowseFailed)

End Function
```

frmSplash.frm

```
VERSION 5.00
Begin VB.Form frmSplash
   BorderStyle     =   3  'Fixed Dialog
   ClientHeight    =   4710
   ClientLeft      =   45
   ClientTop       =   45
   ClientWidth     =   7455
   ControlBox      =   0   'False
   Icon            =   "frmSplash.frx":0000
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   4710
   ScaleWidth      =   7455
   ShowInTaskbar   =   0   'False
   StartUpPosition =   2  'CenterScreen
   Visible         =   0   'False
   Begin VB.Frame fraMainFrame
      Height          =   4590
      Left            =   45
      TabIndex        =   0
      Top             =   -15
      Width           =   7380
      Begin VB.PictureBox picLogo
         Height          =   2385
         Left            =   510
         Picture         =   "frmSplash.frx":0442
         ScaleHeight     =   2325
         ScaleWidth      =   1755
         TabIndex        =   1
         Top             =   855
         Width           =   1815
      End
      Begin VB.Label lblReserved
         AutoSize        =   -1  'True
         Caption         =   "All Rights Reserved."
         Height          =   195
         Left            =   5520
         TabIndex        =   8
         Top             =   4080
         Width           =   1440
      End
      Begin VB.Label lblCopyright
         AutoSize        =   -1  'True
         Caption         =   "Copyright □ 1998"
         BeginProperty Font
            Name            =   "Arial"
            Size            =   8.25
            Charset         =   0
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   210
         Left            =   5550
         TabIndex        =   7
         Tag             =   "Copyright"
         Top             =   3850
         Width           =   1380
         WordWrap        =   -1  'True
      End
      Begin VB.Label lblCompany
         AutoSize        =   -1  'True
         Caption         =   "Microsoft Corporation"
         BeginProperty Font
            Name            =   "Arial"
            Size            =   8.25
            Charset         =   0
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   210
         Left            =   5460
         TabIndex        =   6
         Tag             =   "Company"
         Top             =   3640
         Width           =   1560
      End
      Begin VB.Label lblRestrictions
         AutoSize        =   -1  'True
         Caption         =   "See License Agreement for Restrictions"
         Height          =   195
         Left            =   460
         TabIndex        =   5
         Top             =   4080
         Width           =   2790
      End
      Begin VB.Label lblProductName
         AutoSize        =   -1  'True
         Caption         =   "StepMaster"
         BeginProperty Font
            Name            =   "Arial"
            Size            =   32.25
            Charset         =   0
            Weight          =   700
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   765
         Left            =   2670
         TabIndex        =   4
         Tag             =   "Product"
         Top             =   1200
         Width           =   3495
      End
```

```
      Begin VB.Label lblVersion
         Alignment       =   1  'Right Justify
         AutoSize        =   -1  'True
         Caption         =   "Version 2.4"
         BeginProperty Font
            Name            =   "Arial"
            Size            =   12
            Charset         =   0
            Weight          =   700
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   285
         Left            =   4800
         TabIndex        =   3
         Tag             =   "Version"
         Top             =   2040
         Width           =   1275
      End
      Begin VB.Label lblWarning
         AutoSize        =   -1  'True
         Caption         =   "Do Not Redistribute"
         BeginProperty Font
            Name            =   "Arial"
            Size            =   8.25
            Charset         =   0
            Weight          =   400
            Underline       =   0   'False
            Italic          =   0   'False
            Strikethrough   =   0   'False
         EndProperty
         Height          =   210
         Left            =   480
         TabIndex        =   2
         Tag             =   "Warning"
         Top             =   3850
         Width           =   1380
      End
   End
End
Attribute VB_Name = "frmSplash"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'  FILE:       frmSplash.frm
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Splash screen for StepMaster
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private Sub Form_Load()
'    lblVersion.Caption = "Version " & App.Major & "." & App.Minor & "."
& App.Revision
    lblProductName.Caption = App.Title
    lblVersion.Caption = GetVersionString

End Sub
```

frmWorkspaceOpen.frm

```
VERSION 5.00
Begin VB.Form frmWorkspaceOpen
   BorderStyle     =   3  'Fixed Dialog
   Caption         =   "Open Workspace"
   ClientHeight    =   2550
   ClientLeft      =   45
   ClientTop       =   330
   ClientWidth     =   4695
   LinkTopic       =   "Form1"
   LockControls    =   -1  'True
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   2550
   ScaleWidth      =   4695
   ShowInTaskbar   =   0   'False
   StartUpPosition =   1  'CenterOwner
   Begin VB.CommandButton cmdOK
      Caption         =   "OK"
      Default         =   -1  'True
      Height          =   375
      Left            =   2160
      TabIndex        =   2
      Top             =   2040
      Width           =   1095
   End
   Begin VB.CommandButton cmdCancel
      Cancel          =   -1  'True
      Caption         =   "Cancel"
      Height          =   375
      Left            =   3360
      TabIndex        =   3
      Top             =   2040
      Width           =   1095
   End
   Begin VB.ListBox lstWorkspaces
      Height          =   1425
      ItemData        =   "frmWorkspaceOpen.frx":0000
      Left            =   240
      List            =   "frmWorkspaceOpen.frx":0002
      MultiSelect     =   2  'Extended
      TabIndex        =   1
      Top             =   480
```

```
      Width           =   4215
   End
   Begin VB.Label lblWorkspaces
      AutoSize        =   -1  'True
      Caption         =   "Workspace"
      Height          =   195
      Left            =   240
      TabIndex        =   0
      Top             =   120
      Width           =   825
   End
End
Attribute VB_Name = "frmWorkspaceOpen"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = False
Attribute VB_PredeclaredId = True
Attribute VB_Exposed = False
'  FILE:       frmWorkspaceOpen.frm
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Used to display a list of all workspaces in a
'              workspace definition file for Open, Import, Export
'              and Run (in SMRunOnly) workspace
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
#If RUN_ONLY Then
    Private WithEvents cRunOnlyInst As cRunOnly
Attribute cRunOnlyInst.VB_VarHelpID = -1

Private Sub cRunOnlyInst_Done()
    ShowFree
End Sub

Private Sub Run()
    Dim iIndex As Integer

    Set cRunOnlyInst = New cRunOnly

    For iIndex = 0 To Me.lstWorkspaces.ListCount - 1
        If Me.lstWorkspaces.Selected(iIndex) Then
            ShowBusy
            cRunOnlyInst.WorkspaceId =
Me.lstWorkspaces.ItemData(Me.lstWorkspaces.ListIndex)
            cRunOnlyInst.WspName =
Me.lstWorkspaces.List(Me.lstWorkspaces.ListIndex)
            cRunOnlyInst.RunWsp
        End If
    Next iIndex

End Sub

#End If

Private Sub cmdCancel_Click()

    Unload Me

End Sub

Private Sub cmdOK_Click()

#If RUN_ONLY Then
    Call Run
#Else
    Call WorkspaceOpenOk
#End If

End Sub


Private Sub lstWorkspaces_DblClick()

    ' A double click on the workspaces list box is considered
    ' equivalent to selecting an item in the list and then selecting
    ' the OK command
    Call cmdOK_Click

End Sub
```

IteratorCommon.bas

```
Attribute VB_Name = "IteratorCommon"
'  FILE:       IteratorCommon.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains functionality common across StepMaster and
'              SMRunOnly, pertaining to iterators
'              Specifically, functions to read iterators records
'              in the workspace, load them in an array and so on.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "IteratorCommon."

Public Const gintMinIteratorSequence As Integer = 0

Public Sub RangeComplete(vntIterators As Variant)
    ' This is a debug procedure
    ' Checks if the from, to and step values are present in
    ' the array

    Dim bReset As Byte
```

```
    Dim bShift As Byte
    Dim lngIndex As Long

    ' Set the three lowest order bits to 1
    bReset = 7

    BugAssert IsArray(vntIterators) And Not IsEmpty(vntIterators), _
            "Iterators not specified!"

    For lngIndex = LBound(vntIterators) To _
            UBound(vntIterators)
        bShift = 1
        bShift = bShift * (2 ^ (vntIterators(lngIndex).IteratorType -
1))

        bReset = bReset Xor bShift
    Next lngIndex

    ' Assert that all the elements are present
    BugAssert bReset = 0, "Range not completely specified!"

End Sub
Public Sub LoadIteratorsForWsp(cStepsCol As cArrSteps, _
        ByVal lngWorkspaceId As Long, rstStepsInWsp As Recordset)
    ' Initializes the step records in with all the iterator
    ' values for each step

    Dim recIterators As Recordset

    On Error GoTo LoadIteratorsForWspErr

#If QUERY_ALL Then
    Dim dtStart As Date

    dtStart = Now
    Set recIterators = ReadWspIterators(lngWorkspaceId)

    Call LoadIteratorsArray(cStepsCol, recIterators)

    recIterators.Close

    BugMessage "QueryAll Read + load took: " & CStr(DateDiff("s",
dtStart, Now))

#Else
    Dim dtStart As Date
    Dim qyIt As DAO.QueryDef
    Dim sSql As String

    dtStart = Now
    If rstStepsInWsp.RecordCount = 0 Then
        Exit Sub
    End If

    ' This method has the advantage that if the steps are queried right,
everything else follows
    sSql = "Select step_id, version_no, type, iterator_value, " & _
        " sequence_no " & _
        " from iterator_values " & _
        " where step_id = [s_id] " & _
        " and version_no = [ver_no] "

    ' Order the iterators by sequence within a step
    sSql = sSql & " order by sequence_no "

    Set qyIt = dbsAttTool.CreateQueryDef(gstrEmptyString, sSql)
    rstStepsInWsp.MoveFirst

    While Not rstStepsInWsp.EOF

        qyIt.Parameters("s_id").Value = rstStepsInWsp!step_id
        qyIt.Parameters("ver_no").Value = rstStepsInWsp!version_no

        Set recIterators = qyIt.OpenRecordset(dbOpenSnapshot)

        Call LoadIteratorsArray(cStepsCol, recIterators)
        recIterators.Close

        rstStepsInWsp.MoveNext
    Wend

    qyIt.Close

    BugMessage "Query step at a time Read + load took: " &
CStr(DateDiff("s", dtStart, Now))

#End If

    Exit Sub

LoadIteratorsForWspErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadIteratorsForWsp"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, _
            gstrSource, _
            LoadResString(errLoadRsInArrayFailed)

End Sub
Private Function ReadWspIterators(ByVal lngWorkspaceId As Long) As
Recordset

    ' This function will return a recordset that is populated
    ' with the iterators for all the steps in a given workspace

    Dim recIterators As Recordset
    Dim qyIt As DAO.QueryDef
    Dim strSql As String

    On Error GoTo ReadWspIteratorsErr
    gstrSource = mstrModuleName & "ReadWspIterators"
```

```
    strSql = "Select i.step_id, i.version_no, " & _
        " i.type, i.iterator_value, " & _
        " i.sequence_no " & _
        " from iterator_values i, att_steps a " & _
        " where i.step_id = a.step_id " & _
        " and i.version_no = a.version_no " & _
        " and a.workspace_id = [w_id] " & _
        " and a.archived_flag = [archived] "

    ' Find the highest X-component of the version number
    strSql = strSql & " AND cint( mid( a.version_no, 1,
instr( a.version_no, " & gstrDQ & gstrVerSeparator & gstrDQ & " ) -
1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS d " & _
        " WHERE a.step_id = d.step_id ) "

    ' Find the highest Y-component of the version number for the highest
X-component
    strSql = strSql & " AND cint( mid( a.version_no, instr( a.version_no,
" & gstrDQ & gstrVerSeparator & gstrDQ & " ) + 1 ) ) = " & _
        " ( select max( cint( mid( version_no, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) + 1 ) ) ) " & _
        " from att_steps AS b " & _
        " Where a.step_id = b.step_id " & _
        " AND cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS c " & _
        " WHERE a.step_id = c.step_id ) ) "


    ' Order the iterators by sequence within a step
    strSql = strSql & " order by i.step_id, i.sequence_no "

    Set qyIt = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)
    qyIt.Parameters("w_id").Value = lngWorkspaceId
    qyIt.Parameters("archived").Value = False

    Set recIterators = qyIt.OpenRecordset(dbOpenSnapshot)

    qyIt.Close
    Set ReadWspIterators = recIterators

    Exit Function

ReadWspIteratorsErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errReadDataFailed, _
            gstrSource, LoadResString(errReadDataFailed)

End Function
```

```
Private Sub LoadIteratorsArray(cStepsCol As cArrSteps, _
        recIterators As Recordset)
    ' Initializes the step records with the iterators for
    ' the step

    Dim cNewIt As cIterator
    Dim cStepRec As cStep
    Dim lngStepId As Long

    On Error GoTo LoadIteratorsArrayErr
    gstrSource = mstrModuleName & "LoadIteratorsArray"

    If recIterators.RecordCount = 0 Then
        Exit Sub
    End If

    recIterators.MoveFirst
    While Not recIterators.EOF
        Set cNewIt = New cIterator

        lngStepId = CLng(ErrorOnNullField(recIterators, "step_id"))
        If Not cStepRec Is Nothing Then
            If cStepRec.StepId <> lngStepId Then
                Set cStepRec = cStepsCol.QueryStep(lngStepId)
            End If
        Else
            Set cStepRec = cStepsCol.QueryStep(lngStepId)
        End If

        ' Initialize iterator values
        cNewIt.IteratorType = CInt(ErrorOnNullField(recIterators,
"type"))
        cNewIt.Value = CStr(ErrorOnNullField(recIterators,
"iterator_value"))
        cNewIt.SequenceNo = CInt(ErrorOnNullField(recIterators,
"sequence_no"))

        ' Add this record to the array of iterators
        cStepRec.LoadIterator cNewIt

        Set cNewIt = Nothing
        recIterators.MoveNext
    Wend

    Exit Sub

LoadIteratorsArrayErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadIteratorsArray"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, _
            gstrSource, _
            LoadResString(errLoadRsInArrayFailed)

End Sub
```

MsgConfirm.bas

```
Attribute VB_Name = "MsgConfirm"
'  FILE:       MsgConfirm.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains constants for confirmation messages that
'              will be displayed by StepMaster
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' A public enum containing the codes for all the confirmation
' messages that will be used by the project - each of the codes
' has the prefix, con
Public Enum conConfirmMsgCodes
    conWspDelete = 2000
    conSave
    conStopRun
    conSaveConnect
    conSaveDB
End Enum

' A public enum containing the titles for all the confirmation
' messages that will be used by the project - each of the codes
' has the prefix, cont - most confirmation message codes will
' have a corresponding title code in here
Public Enum conConfirmMsgTitleCodes
    contWspDelete = 3000
    contSave
    contStopRun
    contSaveConnect
    contSaveDB
End Enum
```

OpenFiles.bas

```
Attribute VB_Name = "OpenFiles"
'  FILE:       OpenFiles.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module holds a list of all files that have been
'              opened by the project. This module is needed since there
'              is no way to share static data between different
instances
'              of a class.
'              Many procedure in this module do not do any error
handling -
'              this is 'coz it is also used by procedures that log error
'              messages and any error handler will erase the collection
'              of errors!

'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit


' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = ".OpenFiles."

Private mOpenFiles As cNodeCollections

Private Const mstrTempDir As String = "\Temp\"

' The maximum number of temporary files that we can create in a
' session
Private Const mlngMaxFileIndex As Long = 999999
Private Const mstrFileIndexFormat As String = "000000"
Private Const mstrTempFilePrefix As String = "SM"
Private Const mstrTempFileSuffix As String = ".cmd"

Private Const merrFileNotFound As Long = 76
Private Function GetFileHandle(strFileName) As cFileInfo

    Dim lngIndex As Long
    Dim blnFileOpen As Boolean

    If Not mOpenFiles Is Nothing Then

        blnFileOpen = False
        For lngIndex = 0 To mOpenFiles.Count - 1
            If mOpenFiles(lngIndex).FileName = strFileName Then
                blnFileOpen = True
                Exit For
            End If
        Next lngIndex

        If blnFileOpen Then
            Set GetFileHandle = mOpenFiles(lngIndex)
        Else
            Set GetFileHandle = Nothing
        End If
    Else
        Set GetFileHandle = Nothing
    End If

End Function

Private Function GetTempFileDir() As String

    Dim strTempFileDir As String

    On Error GoTo GetTempFileDirErr
```

```
    strTempFileDir = gstrProjectPath & mstrTempDir

    If StringEmpty(Dir$(strTempFileDir, vbDirectory)) Then
        MkDir strTempFileDir
    End If

    GetTempFileDir = strTempFileDir

    Exit Function

GetTempFileDirErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "GetTempFileDir"
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, gstrSource, _
            LoadResString(errProgramError)

End Function
Public Function MakePathValid(strFileName As String) As String
    ' Checks if the passed in file path is valid

    Dim strFileDir As String
    Dim strTempDir As String
    Dim strTempFile As String
    Dim intPos As Integer
    Dim intStart As Integer

    On Error GoTo MakePathValidErr
    gstrSource = mstrModuleName & "MakePathValid"

    strTempFile = strFileName
    intPos = InstrR(strFileName, gstrFileSeparator)

    If intPos > 0 Then
        strFileDir = Left$(strTempFile, intPos - 1)
        If StringEmpty(Dir$(strFileDir, vbDirectory)) Then
            ' Loop through the entire path starting at the root
            ' since Mkdir can create only one level of sub-directory
            ' at a time
            intStart = InStr(strFileDir, gstrFileSeparator)

            Do While strTempDir <> strFileDir

                If intStart > 0 Then
                    strTempDir = Left$(strFileDir, intStart - 1)
                Else
                    strTempDir = strFileDir
                End If

                If StringEmpty(Dir$(strTempDir, vbDirectory)) Then
                    ' If the specified directory doesn't exist, try to
                    ' create it.
                    MkDir strTempDir
                Else
                    ' The directory exists - go to it's sub-directory
                End If
                intStart = InStr(intStart + 1, strFileDir,
gstrFileSeparator)
            Loop

            ' Sanity check
            If StringEmpty(Dir$(strFileDir, vbDirectory)) Then
                ' We were unable to create the file directory
                ShowError errCreateDirectoryFailed, gstrSource, _
                        strFileDir, DoWriteError:=False
                MakePathValid = gstrEmptyString
            Else
                MakePathValid = strTempFile
            End If
        Else
            ' The specified directory exists - we should be able
            ' to create the output file in it
            MakePathValid = strTempFile
        End If
    Else
        ' The user has only specified a filename - VB will try
        ' to create it in the current directory
        MakePathValid = strTempFile
    End If

    Exit Function

MakePathValidErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "MakePathValid"
    ' Log the filename for debug
    Call WriteError(errInvalidFile, gstrSource, strTempFile)
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, gstrSource, _
            LoadResString(errProgramError)

End Function
Public Function OpenFileSM(strFileName As String) As Integer
    Dim intHFile As Integer
    Dim NewFileInfo As cFileInfo

    On Error GoTo OpenFileSMErr
    gstrSource = mstrModuleName & "OpenFileSM"

    If StringEmpty(strFileName) Then
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidFile, gstrSource, _
                LoadResString(errInvalidFile)
    End If

    If mOpenFiles Is Nothing Then
        Set mOpenFiles = New cNodeCollections
    End If

    Set NewFileInfo = GetFileHandle(strFileName)
```

```
    If NewFileInfo Is Nothing Then
        ' The file has not been opened yet

        ' If the filename has not been initialized, do not
        ' attempt to open it
        strFileName = MakePathValid(strFileName)

        If strFileName <> gstrEmptyString Then
            intHFile = FreeFile
            Open strFileName For Output Shared As intHFile

            Set NewFileInfo = New cFileInfo
            NewFileInfo.FileHandle = intHFile
            NewFileInfo.FileName = strFileName
            mOpenFiles.Load NewFileInfo
        Else
            ' Either the directory was invalid or s'thing failed
            ' Display the error to the user instead of trying
            ' to log to the file
            ShowError errInvalidFile, gstrSource, strFileName, _
                    DoWriteError:=False
            intHFile = 0
        End If
    Else
        intHFile = NewFileInfo.FileHandle
    End If

    OpenFileSM = intHFile

    Exit Function

OpenFileSMErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' The Open command failed for some reason - write an error
    ' and let the calling function handle the error
    ShowError errInvalidFile, gstrSource, strFileName, _
            DoWriteError:=False
    OpenFileSM = 0

End Function
Public Function CreateTempFile() As String

    Dim strTempFileDir As String
    Dim strTempFileName As String

    Static lngLastFileIndex As Long

    On Error GoTo CreateTempFileErr

    strTempFileDir = GetTempFileDir()

    Do
        If lngLastFileIndex = mlngMaxFileIndex Then
            On Error GoTo 0
                Err.Raise vbObjectError + errMaxTempFiles, gstrSource, _
                        LoadResString(errMaxTempFiles)
            End If

            lngLastFileIndex = lngLastFileIndex + 1
            strTempFileName = mstrTempFilePrefix & _
                    Format$(lngLastFileIndex, mstrFileIndexFormat) & _
                    mstrTempFileSuffix

            If Not StringEmpty(Dir$(strTempFileDir & strTempFileName)) Then
                ' Remove any files left over from a previous run,
                ' if they still exist
                Kill strTempFileDir & strTempFileName
            End If

        ' Looping in case the file delete doesn't go through for
        ' some reason
        Loop While Not StringEmpty(Dir$(strTempFileDir & strTempFileName))

        CreateTempFile = GetShortName(strTempFileDir)
        CreateTempFile = CreateTempFile & strTempFileName

        Exit Function

    CreateTempFileErr:
        ' Log the error code raised by Visual Basic
        Call LogErrors(Errors)
        gstrSource = gstrSource & "CreateTempFile"
        On Error GoTo 0
        Err.Raise vbObjectError + errProgramError, gstrSource, _
                LoadResString(errProgramError)

    End Function
    Public Sub CloseFileSM(strFileName As String)
        Dim FileToClose As cFileInfo

        If Not mOpenFiles Is Nothing Then

            ' Get the handle to the open file, if it exists
            Set FileToClose = GetFileHandle(strFileName)

            If Not FileToClose Is Nothing Then
                Close FileToClose.FileHandle

                ' Remove the file info from the collection of open files
                mOpenFiles.Unload FileToClose.Position
            End If
        End If

    End Sub
    Public Sub CloseOpenFiles()
        Dim lIndex As Long

        If Not mOpenFiles Is Nothing Then
            For lIndex = mOpenFiles.Count - 1 To 0
                CloseFileSM (mOpenFiles(lIndex).FileName)
```

```
        Next lIndex
    End If

End Sub
```

ParameterCommon.bas

```
Attribute VB_Name = "ParameterCommon"
'  FILE:       ParameterCommon.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains functionality common across StepMaster and
'              SMRunOnly, pertaining to parameters
'              Specifically, functions to load parameter records
'              in an array.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "ParameterCommon."

Public Sub LoadRecordsetInParameterArray(rstWorkSpaceParameters As
Recordset, _
    cParamCol As cArrParameters)

    Dim cNewParameter As cParameter

    On Error GoTo LoadRecordsetInParameterArrayErr

    If rstWorkSpaceParameters.RecordCount = 0 Then
        Exit Sub
    End If

    rstWorkSpaceParameters.MoveFirst
    While Not rstWorkSpaceParameters.EOF

        Set cNewParameter = New cParameter

        ' Initialize parameter values
        cNewParameter.ParameterId = rstWorkSpaceParameters.Fields(0)

        ' Call a procedure to raise an error if mandatory fields are
        ' null.
        cNewParameter.ParameterName = CStr( _
                ErrorOnNullField(rstWorkSpaceParameters,
"parameter_name"))
        cNewParameter.ParameterValue = CheckForNullField( _
                rstWorkSpaceParameters, "parameter_value")
        cNewParameter.WorkspaceId = CStr( _
                ErrorOnNullField(rstWorkSpaceParameters,
FLD_ID_WORKSPACE))
        cNewParameter.ParameterType = CStr( _
                ErrorOnNullField(rstWorkSpaceParameters,
"parameter_type"))
        cNewParameter.Description = CheckForNullField( _
                rstWorkSpaceParameters, "description")

        cParamCol.Load cNewParameter

        Set cNewParameter = Nothing
        rstWorkSpaceParameters.MoveNext
    Wend

    Exit Sub

LoadRecordsetInParameterArrayErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadRecordsetInParameterArray"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, gstrSource, _
            LoadResString(errLoadRsInArrayFailed)
End Sub
```

Public.bas

```
Attribute VB_Name = "Public"
'  FILE:       Public.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module contains all the public constants for this
project
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Const gsVersion01 As String = "0.1"
Public Const gsVersion10 As String = "1.0"
Public Const gsVersion21 As String = "2.1"
Public Const gsVersion23 As String = "2.3"
Public Const gsVersion24 As String = "2.4"
Public Const gsVersion241 As String = "2.4.1"
Public Const gsVersion242 As String = "2.4.2"
Public Const gsVersion243 As String = "2.4.3"
Public Const gsVersion25 As String = "2.5"
Public Const gsVersion251 As String = "2.5.1"
Public Const gsVersion253 As String = "2.5.3"
Public Const gsVersion254 As String = "2.5.4"
Public Const gsVersion255 As String = "2.5.5"
Public Const gsVersion As String = gsVersion255

' The same form is used for the creation of new nodes and
' updates to existing nodes (where each node can be a parameter,
' global step, etc.) A tag is set on each flag is used to indicate
' whether it is being called in the insert or update mode. The
```

```
' constants for these modes are defined below
Public Const gstrInsertMode = "Insert"
Public Const gstrUpdateMode = "Update"
Public Const gstrPropertiesMode = "View"

Public Const gstrEmptyString = ""
Public Const gstrSQ = "'"
Public Const gstrDQ = """"
Public Const gstrVerSeparator = "."
Public Const gstrBlank = " "

' Constants used to indicate type of node being processed
' The constants for the different objects correspond to the
' indexes in the menu control arrays (for both the main and popup
' menus) that are used to create new objects. That way we can
' use the index passed in by the click event to determine the
' type of node being processed
Public Const gintWorkspace = 1

' Decided to leave it here after some debate over whether it
' actually belongs in the cStep class definition
Public Enum gintStepType
    gintGlobalStep = 3
    gintManagerStep
    gintWorkerStep
End Enum

Public Const gintRunManager = 6
Public Const gintRunWorker = 7

Public Enum gintParameterNodeType
    gintParameter = 8
    gintNodeParamConnection
    gintNodeParamExtension
    gintNodeParamBuiltIn
End Enum

' Leave some constants free for newer types of parameters (?)
Public Const gintConnectionDtl = 15

Public Enum gintLabelNodeType
    gintGlobalsLabel = 21
    gintParameterLabel
    gintParamConnectionLabel
    gintParamExtensionLabel
    gintParamBuiltInLabel
    gintConnDtlLabel
    gintGlobalStepLabel
    gintStepLabel
End Enum

Public Enum ConnectionType
    ConnTypeStatic = 1
    ConnTypeDynamic
End Enum

Public Const giDefaultConnType As Integer = ConnTypeStatic

' The constants defined below are used to identify the different
' tabs. If any more step properties and thereby tabs are added
' to the tabbed dialog on the Step Properties form, they should
' be defined here and accessed in the code only using these
' pre-defined constants
' Note: These constants will mainly be used by the functions that
' initialize, customize and display the Step Properties form
Public Const gintDefinition = 0
Public Const gintExecution = 1
Public Const gintMgrDefinition = 2
Public Const gintPreExecutionSteps = 3
Public Const gintPostExecutionSteps = 4
Public Const gintFileLocations = 5

' These constants correspond to the index values in the imagelist
' associated with the tree view control. The imagelist contains
' the icons that will be displayed for each node.
Public Enum TreeImages
    gintImageWorkspaceClosed = 1
    gintImageWorkspaceOpen
    gintImageLabelClosed
    gintImageLabelOpen
    gintImageManagerClosedDis
    gintImageManagerClosedEn
    gintImageManagerOpenDis
    gintImageManagerOpenEn
    gintImageWorkerDis
    gintImageWorkerEn
    gintImageGlobalClosed
    gintImageGlobalOpen
    gintImageParameter
    gintImageRun
    gintImagePending
    gintImageStop
    gintImageDisabled
    gintImageAborted
    gintImageFailed
End Enum

' Public variable used to indicate the name of the function
' that raises an error
Public gstrSource As String

' Public instances of the different collections
Public gcParameters As cArrParameters
Public gcSteps As cArrSteps
Public gcConstraints As cArrConstraints
Public gcConnections As cConnections
Public gcConnDtls As cConnDtls

' Public constants for the index values of the different toolbar
' options. Will be used while dynamically enabling/disabling
' these options.
Public Const tbNew = 1
```

```
Public Const tbOpen = 2
Public Const tbSave = 3

Public Const tbCut = 5
Public Const tbCopy = 6
Public Const tbPaste = 7
Public Const tbDelete = 8

Public Const tbProperties = 10
Public Const tbRun = 11
Public Const tbStop = 12

' The initial version #
Public Const gstrMinVersion As String = "0.0"
Public Const gstrGlobalParallelism As String = "0"
Public Const gintMinParallelism As Integer = 1
Public Const gintMaxParallelism As Integer = 100

' Constant for the minimum identifier, used for all identifier, viz.
' step, workspace, etc.
Public Const glMinId As Long = 1
Public Const glInvalidId As Long = -1

' A parameter that has a special meaning to Stepmaster
' The system time will be substituted wherever it occurs
' (typically as a part of the error, log ... file names
Public Const gstrTimeStamp As String = "TIMESTAMP"
Public Const gstrEnvVarSeparator = "%"
Public Const gstrFileSeparator = "\"
Public Const gstrUnderscore = "_"

' Constants used by date and time formatting functions
Public Const gsTimeSeparator = ":"
Public Const gsDateSeparator = "-"
Public Const gsMsSeparator = "."
Public Const gsDtFormat = "00"
Public Const gsYearFormat = "0000"
Public Const gsTmFormat = "00"
Public Const gsMSecondFormat = "000"

' Default nothing value for a date variable
Public Const gdtmEmpty As Currency = 0

Public Const FMT_WSP_LOG_FILE As String = "yyyymmdd-hhnnss"

Public gsContCriteria() As String
' Note: Update the initialization of gsExecutionStatus in Initialize()
if the
' InstanceStatus values are modified - also the boundary checks
Public gsExecutionStatus() As String

Public Const gsConnTypeStatic As String = "Static"
Public Const gsConnTypeDynamic As String = "Dynamic"

#If RUN_ONLY Then
Public Const gsCaptionRunWsp As String = "Run Workspace"
#End If

' Valid operations on a cNode object
Public Enum Operation
    QueryOp = 1
    InsertOp = 2
    UpdateOp = 3
    DeleteOp = 4
End Enum
```

RunCommon.bas

```
Attribute VB_Name = "RunCommon"
'  FILE:       RunCommon.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains common functions that are used during the
execution
'              of a workspace.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = ".RunCommon."


Public Function GetInstanceItValue(cInstanceRec As cInstance) As String

    ' Returns the iterator value for the instance, if an
    ' iterator has been defined for it
    Dim cStepIt As cRunColIt
    Dim cRunIterator As cRunItNode

    On Error GoTo GetInstanceItValueErr

    ' Since we create a dummy instance for Disabled and Pending steps,
    ' doesn't make sense to look at their iterators
    If cInstanceRec.Status <> gintDisabled And cInstanceRec.Status <>
gintPending Then
        Set cStepIt = cInstanceRec.Iterators

        If Not StringEmpty(cInstanceRec.Step.IteratorName) Then
            If cStepIt.Count > 0 Then
                Set cRunIterator = cStepIt(0)
                BugAssert cRunIterator.IteratorName =
cInstanceRec.Step.IteratorName, _
                          "The first iterator in the collection is the " &
_
                          "one that has been defined for the step."
                If cRunIterator.IteratorName =
cInstanceRec.Step.IteratorName Then
```

```
                GetInstanceItValue = cRunIterator.Value
            Else
                GetInstanceItValue = gstrEmptyString
            End If
        Else
            GetInstanceItValue = gstrEmptyString
        End If
    Else
        GetInstanceItValue = gstrEmptyString
    End If

    Exit Function

GetInstanceItValueErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    gstrSource = mstrModuleName & "GetInstanceItValue"
    Err.Raise vbObjectError + errProgramError, gstrSource, _
            LoadResString(errProgramError)

End Function
```

RunInstHelper.bas

```
Attribute VB_Name = "RunInstHelper"
'  FILE:       RunInstHelper.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module contains helper procedures that are called by
'              cRunInst.cls
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "RunInstHelper."

' Should be equal to the number of steps defined in cRunInst.cls
Public Const glngNumConcurrentProcesses As Long = 99
Public Const gintBitsPerByte = 8
Public Function AnyStepRunning(cFreeSteps As cVectorLng, arrFree() As
Byte) As Boolean

    Dim lngIndex As Long
    Dim intPosInByte As Integer
    Dim lngTemp As Long

    ' Check if there are any running instances to wait for
    If cFreeSteps.Count <> glngNumConcurrentProcesses Then

        ' For every free step, reset the corresponding element
        ' in the byte array to 0
        For lngIndex = 0 To cFreeSteps.Count - 1

            lngTemp = cFreeSteps(lngIndex) \ gintBitsPerByte
            intPosInByte = cFreeSteps(lngIndex) Mod gintBitsPerByte

            arrFree(lngTemp) = arrFree(lngTemp) Xor 2 ^ intPosInByte
        Next lngIndex

        AnyStepRunning = False

        ' Check if we have a non-zero bit in the byte array
        For lngIndex = LBound(arrFree) To UBound(arrFree) Step 1
            If arrFree(lngIndex) <> 0 Then
                ' We are waiting for a step to complete
                AnyStepRunning = True
                Exit For
            End If
        Next lngIndex

    Else
        AnyStepRunning = False
    End If

End Function


Public Function OrderConstraints(vntTempCons() As Variant, _
        intConsType As ConstraintType) As Variant
    ' Returns a variant containing all the constraint records in the
order
    ' in which they should be executed

    Dim vntTemp As Variant
    Dim lngOuter As Long
    Dim lngInner As Long
    Dim cTempConstraint As cConstraint
    Dim cConstraints() As cConstraint
    Dim lngConsCount As Long
    Dim lngLbound As Long
    Dim lngUbound As Long
    Dim lngStep As Long

    On Error GoTo OrderConstraintsErr

    If intConsType = gintPreStep Then
        ' Since we are travelling up and we need to execute the
constraints
        ' for the top-level steps first, reverse the order that they
        ' have been stored in the array
        lngLbound = UBound(vntTempCons)
        lngUbound = LBound(vntTempCons)
```

```
        lngStep = -1
    Else
        lngLbound = LBound(vntTempCons)
        lngUbound = UBound(vntTempCons)
        lngStep = 1
    End If

    lngConsCount = 0

    For lngOuter = lngLbound To lngUbound Step lngStep
        vntTemp = vntTempCons(lngOuter)

        If Not IsEmpty(vntTemp) Then
            ' Each of the elements is an array
            For lngInner = LBound(vntTemp) To UBound(vntTemp) Step 1
                If Not IsEmpty(vntTemp(lngInner)) Then
                    Set cTempConstraint = vntTemp(lngInner)

                    If Not cTempConstraint Is Nothing Then
                        ReDim Preserve cConstraints(lngConsCount)
                        Set cConstraints(lngConsCount) = cTempConstraint
                        lngConsCount = lngConsCount + 1
                    End If
                End If
            Next lngInner
        End If
    Next lngOuter

    ' Set the return value of the function to the array of
    ' constraints that has been built above
    If lngConsCount = 0 Then
        OrderConstraints = Empty
    Else
        OrderConstraints = cConstraints()
    End If

    Exit Function

OrderConstraintsErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errExecInstanceFailed, _
            mstrModuleName, LoadResString(errExecInstanceFailed)

End Function
```

ShellSM.bas

```
Attribute VB_Name = "ShellSM"
'  FILE:       ShellSM.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module contains a function that creates a process
and
'              waits for it to complete.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Function SyncShell(CommandLine As String, Optional Timeout As
Long, _
    Optional WaitForInputIdle As Boolean) As Boolean

    Dim proc As PROCESS_INFORMATION
    Dim Start As STARTUPINFO
    Dim ret As Long
    Dim nMilliseconds As Long

    BugMessage "Executing: " & CommandLine
    If Timeout > 0 Then
        nMilliseconds = Timeout
    Else
        nMilliseconds = INFINITE
    End If

    'Initialize the STARTUPINFO structure:
    Start.cb = Len(Start)
    Start.dwFlags = STARTF_USESHOWWINDOW
    Start.wShowWindow = SW_SHOWMINNOACTIVE

    'Start the shelled application:
    CreateProcessA 0&, CommandLine, 0&, 0&, 1&, _
        NORMAL_PRIORITY_CLASS, 0&, 0&, Start, proc

    If WaitForInputIdle Then
        'Wait for the shelled application to finish setting up its UI:
        ret = InputIdle(proc.hProcess, nMilliseconds)
    Else
        'Wait for the shelled application to terminate:
        ret = WaitForSingleObject(proc.hProcess, nMilliseconds)
    End If

    CloseHandle proc.hProcess

    'Return True if the application finished. Otherwise it timed out or
erred.
    SyncShell = (ret = WAIT_OBJECT_0)
End Function
```

SMErr.bas

```
Attribute VB_Name = "SMErr"
'  FILE:       SMErr.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
```

```
'  PURPOSE:    This module contains error code for all the errors that
are
'              raised by StepMaster.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' A public enum containing the codes for all the error
' messages that will be displayed by the project - each
' of the codes has the prefix, err
Public Enum errErrorConstants
    errParameterIdInvalid = 1000
    errParameterNameMandatory
    errParameterInsertFailed
    errStepLabelOrTextOrFileRequired
    errMandatoryNodeTextMissing
    errParameterUpdateFailed
    errDupConnDtlName
    errDummy14
    errContCriteriaMandatory
    errContCriteriaNullForGlobal
    errContCriteriaInvalid = 1010
    errParamSeparatorMissing
    errStepTextOrTextFileMandatory
    errStepTextOrFile
    errEnabledFlagFalseForGlobal
    errEnabledFlagLetFailed
    errDegParallelismNullForGlobal
    errInvalidDegParallelism
    errExecutionMechanismInvalid
    errExecutionMechanismLetFailed
    errStepLevelNull = 1020
    errStepLevelZeroForGlobal
    errStepLevelLetFailed
    errRowCountNumeric
    errConnNameInvalid
    errTimeoutNumeric
    errResetConnPropertiesFailed
    errFailureThresholdNumeric
    errConnectionUpdateFailed
    errGlobalRunMethodMandatory
    errGlobalRunMethodNull = 1030
    errGlobalRunMethodInvalid
    errGlobalRunMethodLetFailed
    errNoTrueOption
    errGetOptionFailed
    errSetEnabled
    errStepLabelTextAndFileNull
    errInvalidNodeType
    errSetOptionFailed
    errQueryParameterFailed
    errParamNotFound = 1040
    errQueryStepFailed
    errStepNotFound
    errReadFromScreenFailed
    errCopyPropertiesToFormFailed
    errMandatoryFieldNull
    errUnableToCheckNull
    errUpgradeFailed
    errFindStepSequenceFailed
    errFindParentStepIdFailed
    errCircularReference = 1050
    errAddStepFailed
    errModifyStepFailed
    errDeleteColumnFailed
    errFindPositionFailed
    errDeleteStepFailed
    errRunExistsForStepFailed
    errCreateNewParentFailed
    errInsertNewStepVersionFailed
    errCreateNewStepFailed
    errDuplicateParameterName = 1060
    errCheckDupParameterNameFailed
    errConstraintTypeInvalid
    errConstraintTypeLetFailed
    errAddConstraintFailed
    errWorkspaceIdMandatory
    errInvalidWorkspaceData
    errGetWorkspaceDetailsFailed
    errNoWorkspaceLoaded
    errWorkspaceAlreadyOpen
    errDuplicateWorkspaceName = 1070
    errWorkspaceNameMandatory
    errWorkspaceNameSetFailed
    errWorkspaceIdInvalid
    errWorkspaceIdSetFailed
    errWorkspaceInsertFailed
    errWorkspaceDeleteFailed
    errWorkspaceUpdateFailed
    errInvalidFile
    errCheckWorkspaceOpenFailed
    errWriteFailed = 1080
    errDeleteParameterRecordFailed
    errUnableToLogOutput
    errDeleteDBRecordFailed
    errRunExistsForWorkspaceFailed
    errClearHistoryFailed
    errCreateDBFailed
    errImportWspFailed
    errStepModifyFailed
    errStepDeleteFailed
    errDummy3 = 1090
    errUnableToGetWorkspace
    errInvalidNode
    errUnableToRemoveSubtree
    errSetFileNameFailed
    errStepTypeInvalid
    errObjectMandatory
    errBuiltInUpdateOnly
    errInvalidStep
    errTypeOfStepFailed
    errGetParentKeyFailed = 1100
```

```
errLabelTextAndFileCheckFailed
errStepTextAndFileNull
errTextOrFileCheckFailed
errParentStepManager
errDeleteSubStepsFailed
errWorkspaceNameDuplicateFailed
errNewConstraintVersionFailed
errDeleteStepConstraintsFailed
errOldVersionMandatory
errLoadConstraintsInListFailed = 1110
errLoadGlobalStepsFailed
errDeleteConstraintFailed
errUpdateConstraintFailed
errConstraintIdInvalid
errConstraintIdSetFailed
errGlobalStepIdInvalid
errGlobalStepIdSetFailed
errUpdateVersionFailed
errQueryAdjacentConsFailed
errConstraintNotFound = 1120
errQueryConstraintFailed
errSetDBBeforeLoad
errLoadDataFailed
errLoadRsInArrayFailed
errConstraintsForStepFailed
errPreConstraintsForStepFailed
errPostConstraintsForStepFailed
errExecuteConstraintMethodFailed
errIdOrKeyMandatory
errInListFailed = 1130
errUnableToWriteChanges
errQuickSortFailed
errCheckParentValidFailed
errLogErrorFailed
errCopyListFailed
errConnected
errVersionMismatch
errStepNodeFailed
errWorkspaceSelectedFailed
errIdentifierSelectedFailed = 1140
errCheckForNullFieldFailed
errInstanceInUse
errSetVisiblePropertyFailed
errExportWspFailed
errMakeKeyValidFailed
errDummy16
errRunApplicationFailed
errStepLabelUnique
errDeleteSingleFile
errMakeIdentifierValidFailed = 1150
errTypeOfNodeFailed
errConstraintCommandFailed
errOpenDbFailed
errInsertNewConstraintsFailed
errLoadPostExecutionStepsFailed
errLoadPreExecutionStepsFailed
errCreateNewNodeFailed
errDeleteNodeFailed
errDisplayPopupFailed
errDisplayPropertiesFailed = 1160
errUnableToCreateNewObject
errDiffFailed
errLoadWorkspaceFailed
errTerminateProcessFailed
errCompareFailed
errCreateConnectionFailed
errShowFormFailed
errAbortFailed
errDeleteParameterFailed
errUpdateViewFailed = 1170
errParameterNewFailed
errCopyNodeFailed
errCutNodeFailed
errCheckObjectValidFailed
errDeleteViewNodeFailed
errMainFailed
errNewStepFailed
errProcessStepModifyFailed
errCustomizeStepFormFailed
errInitializeStepFormFailed = 1180
errInsertStepFailed
errIncVersionYFailed
errIncVersionXFailed
errShowCreateStepFormFailed
errShowStepFormFailed
errStepNewFailed
errUnableToApplyChanges
errUnableToCommitChanges
errGetStepNodeTextFailed
errSelectGlobalRunMethodFailed = 1190
errConnectionNameMandatory
errUpdateStepFailed
errBrowseFailed
errDummy4
errDummy1
errDummy2
errDummy
errUnableToPreviewFile
errCopyWorkspaceFailed
errCopyParameterFailed = 1200
errGetStepTypeAndPositionFailed
errCopyStepFailed
errMandatoryParameterMissing
errDeleteWorkspaceRecordsFailed
errCreateDirectoryFailed
errConfirmDeleteOrMoveFailed
errCreateWorkspaceFailed
errTypeOfObjectFailed
errCreateNodeFailed
errCreateParameterFailed = 1210
errInsertParameterFailed
errCreateStepFailed
```

```
errNoConstraintsCreated
errCopyFailed
errCloneFailed
errCloneGlobalFailed
errCloneWorkerFailed
errCloneManagerFailed
errLetStepTypeFailed
errUnableToCloseWorkspace = 1220
errUnableToModifyWorkspace
errUnableToCreateWorkspace
errAddArrayElementFailed
errUpdateSequenceFailed
errCannotCopySubSteps
errSubStepsFailed
errModifyInArrayFailed
errUpdateParentVersionFailed
errGetNodeTextFailed
errAddToArrayFailed = 1230
errDeleteFromArrayFailed
errQueryIndexFailed
errCreateNewConstraintVersionFailed
errGetRootNodeFailed
errPopulateWspDetailsFailed
errLoadRsInTreeFailed
errAddNodeToTreeFailed
errMaxTempFiles
errMoveFailed
errRootNodeKeyInvalid = 1240
errNextNodeFailed
errBranchWillMove
errMoveBranchInvalid
errCreateIdRecordsetFailed
errIdentifierColumnFailed
errGetIdentifierFailed
errGetStepTypeFailed
errUpdateConstraintSeqFailed
errDelParamsInWspFailed
errDuplicateConnectionName = 1250
errOpenWorkspaceFailed
errShowWorkspaceNewFailed
errShowWorkspaceModifyFailed
errPopulateListFailed
errExploreNodeFailed
errInitializeListNodeFailed
errMakeListColumnsFailed
errRefreshViewFailed
errExploreFailed
errCollapseNodeFailed = 1260
errUnableToProcessListViewClick
errSetEnabledForStepFailed
errDisplayStepFormFailed
errSetEnabledPropertyFailed
errInvalidDB
errDeleteConnectionFailed
errInvalidOperation
errLetOperationFailed
errIdGetFailed
errCommitFailed = 1270
errSaveParametersInWspFailed
errDeleteArrayElementFailed
errSaveWorkspaceFailed
errInitializeFailed
errLoadInArrayFailed
errSaveStepsInWspFailed
errCommitStepFailed
errStepIdGetFailed
errUnloadFromArrayFailed
errValidateFailed = 1280
errTextEnteredFailed
errStepLabelMandatory
errTextAndFileNullForManager
errFailureDetailsNullForMgr
errSetTabOrderFailed
errSaveWspConstraintsFailed
errCommitConstraintFailed
errUnloadStepConstraintsFailed
errUnableToModifyMenu
errConfirmFailed = 1290
errInitSubItemsFailed
errUpdateListNodeFailed
errAddNodeFailed
errLoadListNodeFailed
errAddListNodeFailed
errExecutionFailed
errSetListViewStyleFailed
errSetCheckedFailed
errGetCheckedFailed
errUnableToProcessListViewDblClick = 1300
errDefaultPosition
errShellFailed
errOpenFileFailed
errSetTBar97Failed
errConnectFailed
errApiFailed
errRegEntryInvalid
errParseStringFailed
errConstraintsForWspFailed
errPostConstraintsForWspFailed = 1310
errPreConstraintsForWspFailed
errLoadWspPostExecStepsFailed
errLoadWspPreExecStepsFailed
errLoadConstraintsOnFormFailed
errQueryFailed
errPasteNodeFailed
errShowAllWorkspacesFailed
errMakeFieldValidFailed
errInitializeTree
errRootNodeFailed = 1320
errDirectionInvalid
errUnableToDetListProperty
errUnableToGetListData
errItemNotFound
```

```
    errItemDoesNotExist
    errParamNameInvalid
    errGetParamValueFailed
    errSubValuesFailed
    errStringOpFailed
    errReadWorkspaceDataFailed = 1330
    errUpdateRunDataFailed
    errProgramError
    errUnableToOpenFile
    errLoadRunDataFailed
    errExecuteODBCCommandFailed
    errRunWorkspaceFailed
    errExecuteStepFailed
    errUnableToWriteError
    errRunStepFailed
    errSaveChanges = 1340
    errDragDropFailed
    errInvalidParameter
    errAssignParametersFailed
    errLoadLabelsInTreeFailed
    errInstrRFailed
    errInsertIteratorFailed
    errDeleteIteratorFailed
    errTypeInvalid
    errLoadFailed
    errDeleteFailed = 1350
    errModifyFailed
    errIteratorsFailed
    errInsertFailed
    errUpdateFailed
    errDuplicateIterator
    errSaveFailed
    errReadDataFailed
    errUnloadFailed
    errAddFailed
    errExecuteBranchFailed = 1360
    errRangeNumeric
    errRangeInvalid
    errNextStepFailed
    errUpdateDisplayFailed
    errDateToStringFailed
    errGetElapsedTimeFailed
    errMaxProcessesExceeded
    errInvalidProperty
    errInvalidChild
    errCreateInstanceFailed = 1370
    errInvalidForWorker
    errInstanceOpFailed
    errNavInstancesFailed
    errIterateFailed
    errExecInstanceFailed
    errDupIterator
End Enum
```

SortSM.bas

```
Attribute VB_Name = "SortSM"
'  FILE:       SortSM.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module contains an implementation of QuickSort.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Comment out for case-sensitive sorts
Option Compare Text

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "SortSM."

Private Function Compare(ByVal vntToCompare1 As Variant, _
        ByVal vntToCompare2 As Variant) As Integer

    On Error GoTo CompareErr

    Compare = 0

    If vntToCompare1.SequenceNo < vntToCompare2.SequenceNo Then
        Compare = -1
    ElseIf vntToCompare1.SequenceNo > vntToCompare2.SequenceNo Then
        Compare = 1
    End If

    Exit Function

CompareErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errCompareFailed, _
            gstrSource, _
            LoadResString(errCompareFailed)

End Function

Private Sub Swap(ByRef vntToSwap1 As Variant, _
        ByRef vntToSwap2 As Variant)

    Dim vntTemp As Variant

    On Error GoTo SwapErr

    If IsObject(vntToSwap1) And IsObject(vntToSwap2) Then
        Set vntTemp = vntToSwap1
        Set vntToSwap1 = vntToSwap2
        Set vntToSwap2 = vntTemp
    Else
```

```
        vntTemp = vntToSwap1
        vntToSwap1 = vntToSwap2
        vntToSwap2 = vntTemp
    End If

    Exit Sub

SwapErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errQuickSortFailed, mstrModuleName &
"Swap", _
        LoadResString(errQuickSortFailed)

End Sub

Public Sub QuickSort(vntArray As Variant, _
        Optional ByVal intLBound As Integer, _
        Optional ByVal intUBound As Integer)
    ' Sorts a variant array using Quicksort

    Dim i As Integer
    Dim j As Integer
    Dim vntMid As Variant

    On Error GoTo QuickSortErr

    If IsEmpty(vntArray) Or _
            Not IsArray(vntArray) Then
        Exit Sub
    End If

    ' Set default boundary values for first time through
    If intLBound = 0 And intUBound = 0 Then
        intLBound = LBound(vntArray)
        intUBound = UBound(vntArray)
    End If

    ' BugMessage "Sorting elements " & Str(intLBound) & " and " &
Str(intUBound)

    If intLBound > intUBound Then
        Exit Sub
    End If

    ' Only two elements in this subdivision; exchange if they
    ' are out of order and end recursive calls
    If (intUBound - intLBound) = 1 Then
        If Compare(vntArray(intLBound), vntArray(intUBound)) > 0 Then
            Call Swap(vntArray(intLBound), vntArray(intUBound))
        End If
        Exit Sub
    End If

    ' Set the pivot point
    Set vntMid = vntArray(intUBound)
    i = intLBound
    j = intUBound

    Do
        ' Move in from both sides towards pivot element
        Do While (i < j) And Compare(vntArray(i), vntMid) <= 0
            i = i + 1
        Loop

        Do While (j > i) And Compare(vntArray(j), vntMid) >= 0
            j = j - 1
        Loop

        If i < j Then
            Call Swap(vntArray(i), vntArray(j))
        End If
    Loop While i < j

    ' Since i has been adjusted, swap element i with element,
    ' intUBound
    Call Swap(vntArray(i), vntArray(intUBound))

    ' Recursively call sort array - pass smaller subdivision
    ' first to conserve stack space
    If (i - intLBound) < (intUBound - 1) Then
        ' Recursively sort with adjusted values for upper and
        ' lower bounds
        Call QuickSort(vntArray, intLBound, i - 1)
        Call QuickSort(vntArray, i + 1, intUBound)
    Else
        Call QuickSort(vntArray, i + 1, intUBound)
        Call QuickSort(vntArray, intLBound, i - 1)
    End If
    Exit Sub

QuickSortErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errQuickSortFailed, mstrModuleName &
"QuickSort", _
        LoadResString(errQuickSortFailed)

End Sub
```

startup.bas

```
Attribute VB_Name = "Startup"
'  FILE:       Startup.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module contains startup and cleanup functions for
the project.
```

```
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Const LISTVIEW_BUTTON = 14

Public gstrProjectPath As String

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "Startup."

Private Sub Initialize()

    On Error GoTo InitializeErr

    ReDim gsContCriteria(gintOnFailureAbort To gintOnFailureAsk) As
String
    gsContCriteria(gintOnFailureAbort) = "Abort"
    gsContCriteria(gintOnFailureContinue) = "Continue"
    gsContCriteria(gintOnFailureCompleteSiblings) = "Execute sibling
steps and stop"
    gsContCriteria(gintOnFailureAbortSiblings) = "Abort sibling steps
and execute next parent"
    gsContCriteria(gintOnFailureSkipSiblings) = "Skip sibling steps and
execute next parent"
    gsContCriteria(gintOnFailureAsk) = "Ask"

    ReDim gsExecutionStatus(gintDisabled To gintAborted) As String
    gsExecutionStatus(gintDisabled) = "Disabled"
    gsExecutionStatus(gintPending) = "Pending"
    gsExecutionStatus(gintRunning) = "Running"
    gsExecutionStatus(gintComplete) = "Complete"
    gsExecutionStatus(gintFailed) = "Failed"
    gsExecutionStatus(gintAborted) = "Stopped"

#If Not RUN_ONLY Then
    ' Call a procedure to change the style of the toolbar
    ' on the Step Properties form
    Call SetTBar97(frmSteps.tblConstraintCommands)

#End If

    Call InitRunEngine

    Exit Sub

InitializeErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "Initialize"
    Call ShowError(errInitializeFailed)

End Sub
Sub Main()
    On Error GoTo MainErr

    ' Mousepointer should indicate busy
    Call ShowBusy

    ' Display the Splash screen while we carry out some initialization
    frmSplash.Show
    frmSplash.Refresh

    gstrProjectPath = App.Path

    ' Open the database
    If OpenDBFile() = False Then
        Unload frmSplash
        Exit Sub
    End If

#If Not RUN_ONLY Then
    Load frmMain

    ' Enable the Stop Run menu options only when a workspace is
    ' actually running
    Call EnableStop(False)

    ' Clear all application extension menu items
    Call ClearToolsMenu
#End If

    Call Initialize

    ' Mousepointer - ready to accept user input
    Call ShowFree

    ' Unload the Splash screen and display the main form
    Unload frmSplash

#If RUN_ONLY Then
    frmWorkspaceOpen.Caption = gsCaptionRunWsp

    Call ShowWorkspacesInDb(dbsAttTool)
#Else
    frmMain.Show
#End If

    Exit Sub

MainErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Call ShowFree
    Call ShowError(errMainFailed)

End Sub
Private Function OpenDBFile() As Boolean
    Dim sDb As String
```

```
    On Error GoTo OpenDBFileErr

#If RUN_ONLY Then
    ' Always use the registry setting for the run_only mode
    sDb = DefaultDBFile()
#Else
    ' Check if the user has specified the workspace defn. file to open
on the command line
    ' Else, use the registry setting
    sDb = IIf(StringEmpty(Command), DefaultDBFile(), Command)

    If Len(sDb) > 0 Then
        ' Trim off the enclosing double-quotes if any
        If Mid(sDb, 1, 1) = gstrDQ Then
            If Len(sDb) > 1 Then
                sDb = Mid(sDb, 2)
            Else
                sDb = gstrEmptyString
            End If
        End If
    End If

    If Len(sDb) > 0 Then
        If Mid(sDb, Len(sDb), 1) = gstrDQ Then
            If Len(sDb) > 1 Then
                sDb = Mid(sDb, 1, Len(sDb) - 1)
            Else
                sDb = gstrEmptyString
            End If
        End If
    End If
#End If

    ' Open the database
    OpenDBFile = SMOpenDatabase(sDb)

    Exit Function

OpenDBFileErr:
    Call LogErrors(Errors)
    OpenDBFile = False

End Function
Public Sub Cleanup()

    On Error GoTo CleanupErr

    ' Set the mousepointer to indicate Busy
    Call ShowBusy

#If Not RUN_ONLY Then
    ' Close all open workspaces - will also prompt for unsaved
    ' changes
    Call CloseOpenWorkspaces
#End If

    ' Close all open files
    Call CloseOpenFiles

    ' Reset the mousepointer
    Call ShowFree

    Exit Sub

CleanupErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Resume Next

End Sub
```

StepCommon.bas

```
Attribute VB_Name = "StepCommon"
'  FILE:       StepCommon.bas
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Contains functionality common across StepMaster and
'              SMRunOnly, pertaining to steps
'              Specifically, functions to load iterators records
'              in an array, determine the type of step, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "StepCommon."

' Step property constants
Private Const mintMinFailureThreshold As Integer = 1
Public Const gintMinSequenceNo As Integer = 1
Public Const gintMinLevel As Integer = 0
Public Function ValidateParallelism(sParallelism As String, lWorkspace
As Long, _
        Optional ParamsInWsp As cArrParameters = Nothing) As String
    ' Returns the degree of parallelism for the step if the user input
is valid
    Dim sTemp As String

    On Error GoTo ValidateParallelismErr
    gstrSource = mstrModuleName & "ValidateParallelism"

    sTemp = SubstituteParameters(Trim$(sParallelism), lWorkspace,
WspParameters:=ParamsInWsp)

    If Not IsNumeric(sTemp) Then
        ShowError errInvalidDegParallelism
        On Error GoTo 0
```

```
        Err.Raise vbObjectError + errInvalidDegParallelism, gstrSource,
_
                LoadResString(errInvalidDegParallelism)
    Else
        If (CInt(sTemp) < gintMinParallelism) Or (CInt(sTemp) >
gintMaxParallelism) Then
            ShowError errInvalidDegParallelism
            On Error GoTo 0
            Err.Raise vbObjectError + errInvalidDegParallelism,
gstrSource, _
                    LoadResString(errInvalidDegParallelism)
        Else
            ValidateParallelism = Trim$(sParallelism)
        End If
    End If

    Exit Function

ValidateParallelismErr:
    ' Log the error code raised by Visual Basic
    gstrSource = mstrModuleName & "ValidateParallelism"
    If Err.Number = vbObjectError + errSubValuesFailed Then
        ShowError errInvalidDegParallelism
    End If

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInvalidDegParallelism, gstrSource, _
        LoadResString(errInvalidDegParallelism)

End Function

Public Function IsGlobal( _
        Optional ByVal StepClass As cStep = Nothing, _
        Optional ByVal StepRecord As Recordset = Nothing, _
        Optional ByVal StepKey As String = gstrEmptyString, _
        Optional ByVal StepId As Long = 0, _
        Optional StepForm As Form = Nothing) As Boolean

    ' This function contains all the possible checks for whether
    ' a step is global - The check that will be made depends on
    ' the parameter passed in

    Dim cStepRecord As cStep

    If Not StepClass Is Nothing Then
        IsGlobal = StepClass.GlobalFlag
        Exit Function
    End If

    If Not StepRecord Is Nothing Then
        IsGlobal = StepRecord![global_flag]
        Exit Function
    End If

    If Not StringEmpty(StepKey) Then
        IsGlobal = InStr(StepKey, gstrGlobalStepPrefix) > 0
        Exit Function
    End If

    If StepId <> 0 Then
        Set cStepRecord = gcSteps.QueryStep(StepId)
        IsGlobal = cStepRecord.GlobalFlag
        Set cStepRecord = Nothing
        Exit Function
    End If

    If Not StepForm Is Nothing Then
        IsGlobal = (StepForm.lblStepType.Caption = Str(gintGlobalStep))
        Exit Function
    End If

    ' Not a single object was passed in! - raise an error
    On Error GoTo 0
    Err.Raise vbObjectError + errObjectMandatory, _
            mstrModuleName & "IsGlobal", _
            LoadResString(errObjectMandatory)

End Function
Public Function TypeOfStep(Optional ByVal StepClass As cStep = Nothing,
_
        Optional ByVal StepRecord As Recordset = Nothing, _
        Optional ByVal StepKey As String = gstrEmptyString, _
        Optional ByVal StepId As Long = 0, _
        Optional StepForm As Form = Nothing) As Integer
    ' Calls functions to determine the type of step
    ' The check that will be made depends on the parameter passed in

    On Error GoTo TypeOfStepErr

    ' Make the check whether a step is global first - both
    ' worker and global steps have the step text or file name
    ' not null - but only the global step will have the global
    ' flag set
    If IsGlobal(StepClass, StepRecord, StepKey, StepId, StepForm) Then
        TypeOfStep = gintGlobalStep
    ElseIf IsManager(StepClass, StepRecord, StepKey, StepId, StepForm)
Then
        TypeOfStep = gintManagerStep
    ElseIf IsWorker(StepClass, StepRecord, StepKey, StepId, StepForm)
Then
        TypeOfStep = gintWorkerStep
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidStep, _
                mstrModuleName & "TypeOfStep", _
                LoadResString(errInvalidStep)
    End If

    Exit Function

TypeOfStepErr:
```

```
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errTypeOfStepFailed, _
            mstrModuleName & "TypeOfStep", _
            LoadResString(errTypeOfStepFailed)

End Function

Public Function IsStep(intNodeType As Integer) As Boolean
    ' Returns true if the node type corresponds to a global, manager
    ' or worker step
    IsStep = (intNodeType = gintGlobalStep) Or (intNodeType =
gintManagerStep) Or _
        (intNodeType = gintWorkerStep)

End Function

Public Function IsManager(Optional ByVal StepClass As cStep = Nothing, _
        Optional ByVal StepRecord As Recordset = Nothing, _
        Optional ByVal StepKey As String = gstrEmptyString, _
        Optional ByVal StepId As Long = 0, _
        Optional StepForm As Form = Nothing) As Boolean

    ' This function contains all the possible checks for whether
    ' a step is a manager step - The check that will be made depends
    ' on the parameter passed in

    Dim cStepRecord As cStep

    If Not StepClass Is Nothing Then
        IsManager = (StepClass.StepType = gintManagerStep)
        Exit Function
    End If

    If Not StepRecord Is Nothing Then
        IsManager = (IsNull(StepRecord![step_text]) And
IsNull(StepRecord![step_file_name]))
        Exit Function
    End If

    If Not StringEmpty(StepKey) Then
        IsManager = (InStr(StepKey, gstrManagerStepPrefix) > 0)
        Exit Function
    End If

    If StepId <> 0 Then
        Set cStepRecord = gcSteps.QueryStep(StepId)
        IsManager = (cStepRecord.StepType = gintManagerStep)
        Set cStepRecord = Nothing
        Exit Function
    End If

    If Not StepForm Is Nothing Then
        IsManager = (StepForm.lblStepType.Caption =
Str(gintManagerStep))
        Exit Function
    End If

    ' Not a single object was passed in! - raise an error
    On Error GoTo 0
    Err.Raise vbObjectError + errObjectMandatory, _
            "Step.IsManager", _
            LoadResString(errObjectMandatory)

End Function
Public Function IsWorker( _
        Optional ByVal StepClass As cStep = Nothing, _
        Optional ByVal StepRecord As Recordset = Nothing, _
        Optional ByVal StepKey As String = gstrEmptyString, _
        Optional ByVal StepId As Long = 0, _
        Optional StepForm As Form = Nothing) As Boolean

    ' This function contains all the possible checks for whether
    ' a step is a Worker step - The check that will be made depends
    ' on the parameter passed in

    Dim cStepRecord As cStep

    If Not StepClass Is Nothing Then
        IsWorker = (StepClass.StepType = gintWorkerStep)
        Exit Function
    End If

    If Not StepRecord Is Nothing Then
        IsWorker = (Not StepRecord![global_flag] And _
                (Not IsNull(StepRecord![step_text]) Or Not
IsNull(StepRecord![step_file_name])))
        Exit Function
    End If

    If Not StringEmpty(StepKey) Then
        IsWorker = InStr(StepKey, gstrWorkerStepPrefix) > 0
        Exit Function
    End If

    If StepId <> 0 Then
        Set cStepRecord = gcSteps.QueryStep(StepId)
        IsWorker = (cStepRecord.StepType = gintWorkerStep)
        Set cStepRecord = Nothing
        Exit Function
    End If

    If Not StepForm Is Nothing Then
        IsWorker = (StepForm.lblStepType.Caption = Str(gintWorkerStep))
        Exit Function
    End If

    ' Not a single object was passed in! - raise an error
    On Error GoTo 0
    Err.Raise vbObjectError + errObjectMandatory, _
            "Step.IsWorker", _
```

```
            LoadResString(errObjectMandatory)

End Function
Public Function GetStepNodeText(ByVal cStepNode As cStep) As String

    On Error GoTo GetStepNodeTextErr

    ' Returns the string that will be displayed as the text
    ' in the tree view node to the user
    If StringEmpty(cStepNode.StepLabel) Then

        If StringEmpty(cStepNode.StepTextFile) Then

            If StringEmpty(cStepNode.StepText) Then
                ' This should never happen
                On Error GoTo 0
                Err.Raise vbObjectError + errStepLabelTextAndFileNull, _
                    gstrSource, _
                    LoadResString(errStepLabelTextAndFileNull)
            Else
                GetStepNodeText = cStepNode.StepText
            End If
        Else
            GetStepNodeText = cStepNode.StepTextFile
        End If
    Else
        GetStepNodeText = cStepNode.StepLabel
    End If

    Exit Function

GetStepNodeTextErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errGetStepNodeTextFailed, _
            gstrSource, _
            LoadResString(errGetStepNodeTextFailed)

End Function

Public Function LoadRecordsetInStepsArray(rstSteps As Recordset, _
    cStepCol As cArrSteps) As Boolean

    Dim cNewStep As cStep
    Dim cNewGlobal As cGlobalStep
    Dim cNewManager As cManager
    Dim cNewWorker As cWorker

    On Error GoTo LoadRecordsetInStepsArrayErr

    If rstSteps.RecordCount = 0 Then
        Exit Function
    End If

    rstSteps.MoveFirst

    While Not rstSteps.EOF
        ' For fields that should not be null, a procedure is first
        ' called to raise an error if the field is null

        Set cNewStep = New cStep

        cNewStep.StepType = TypeOfStep(StepRecord:=rstSteps)

        If cNewStep.StepType = gintGlobalStep Then
            Set cNewGlobal = New cGlobalStep
            Set cNewStep = cNewGlobal
        ElseIf cNewStep.StepType = gintManagerStep Then
            Set cNewManager = New cManager
            Set cNewStep = cNewManager
        Else
            Set cNewWorker = New cWorker
            Set cNewStep = cNewWorker
        End If

        ' Initialize the global flag first, since subsequent
        ' validations might depend on whether the step is global
        cNewStep.GlobalFlag = CBool(ErrorOnNullField(rstSteps,
"global_flag"))

        ' Initialize step values
        cNewStep.StepId = CLng(ErrorOnNullField(rstSteps, "step_id"))
        cNewStep.VersionNo = CStr(ErrorOnNullField(rstSteps,
"version_no"))

        cNewStep.StepLabel = CheckForNullField(rstSteps, "step_label")
        cNewStep.StepTextFile = CheckForNullField(rstSteps,
"step_file_name")
        cNewStep.StepText = CheckForNullField(rstSteps, "step_text")
        cNewStep.StartDir = CheckForNullField(rstSteps,
"start_directory")

        cNewStep.WorkspaceId = CLng(ErrorOnNullField(rstSteps,
FLD_ID_WORKSPACE))
        cNewStep.ParentStepId = CLng(ErrorOnNullField(rstSteps,
"parent_step_id"))
        cNewStep.ParentVersionNo = CStr(ErrorOnNullField(rstSteps,
"parent_version_no"))

        cNewStep.SequenceNo = CInt(ErrorOnNullField(rstSteps,
"sequence_no"))
        cNewStep.StepLevel = CInt(ErrorOnNullField(rstSteps,
"step_level"))
        cNewStep.EnabledFlag = CBool(ErrorOnNullField(rstSteps,
"enabled_flag"))

        ' Initialize the execution details for the step
        cNewStep.DegreeParallelism = CheckForNullField(rstSteps,
"degree_parallelism")
        cNewStep.ExecutionMechanism = CInt(ErrorOnNullField(rstSteps,
"execution_mechanism"))
```

```
        cNewStep.FailureDetails = CheckForNullField(rstSteps,
"failure_details")
        cNewStep.ContinuationCriteria = CInt(ErrorOnNullField(rstSteps,
"continuation_criteria"))

        ' Initialize the output file locations for the step
        cNewStep.OutputFile = CheckForNullField(rstSteps,
"output_file_name")
        ' cNewStep.LogFile = CheckForNullField(rstSteps,
"log_file_name")
        cNewStep.ErrorFile = CheckForNullField(rstSteps,
"error_file_name")

        ' Initialize the iterator name for the step, if any
        cNewStep.IteratorName = CheckForNullField(rstSteps,
"iterator_name")

        ' Add this record to the array of steps
        cStepCol.Load cNewStep

        Set cNewStep = Nothing
        rstSteps.MoveNext
    Wend

    Exit Function

LoadRecordsetInStepsArrayErr:

    LogErrors Errors
    gstrSource = mstrModuleName & "LoadRecordsetInStepsArray"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, gstrSource, _
            LoadResString(errLoadRsInArrayFailed)

End Function
```

F2. Configuration

Att_workspaces

workspace_id	*workspace_name*	*archived_flag*
11	ACID - TPC-H Kit V1.01d (Spec Revision 1.2.0)	0

Workspace_parameters

workspace_id =	11	*parameter_id*	144	*parameter_type =*	0
parameter_name	DUR_RUN_ID				
parameter_valu	51				
workspace_id =	11	*parameter_id*	143	*parameter_type =*	3
parameter_name	OUTPUT_DIR				
parameter_valu	C:\MSTPCH~1.101\OUTPUT\52				
workspace_id =	11	*parameter_id*	142	*parameter_type =*	3
parameter_name	RUN_ID				
parameter_valu	52				
workspace_id =	11	*parameter_id*	141	*parameter_type =*	0
parameter_name	KIT_DIR				
parameter_valu	C:\MSTPCH.101d				
workspace_id =	11	*parameter_id*	140	*parameter_type =*	0
parameter_name	TOOLS_DIR				
parameter_valu	%KIT_DIR%\tools				
workspace_id =	11	*parameter_id*	139	*parameter_type =*	0
parameter_name	SERVER_NAME				
parameter_valu	asama				
workspace_id =	11	*parameter_id*	138	*parameter_type =*	0
parameter_name	NUM_STREAMS				
parameter_valu	9				
workspace_id =	11	*parameter_id*	137	*parameter_type =*	0
parameter_name	DEFAULT_DIR				
parameter_valu	%KIT_DIR%\OUTPUT				

workspace_id =	11	*parameter_id*	136	*parameter_type* =	0
parameter_name	DBNAME				
parameter_valu	tpch3t				

workspace_id =	11	*parameter_id*	135	*parameter_type* =	0
parameter_name	ACID_DIR				
parameter_valu	%KIT_DIR%\ACID				

Connection_dtls

workspace_id	*connection_name_id*	*connection_name*	*connection_string_name*	*connection_type*
11	66	Static_Connection_to_DB_10	DBCONNECTION	1
11	65	Static_Connection_to_DB_9	DBCONNECTION	1
11	64	Static_Connection_to_DB_8	DBCONNECTION	1
11	63	Static_Connection_to_DB_7	DBCONNECTION	1
11	62	Static_Connection_to_DB_6	DBCONNECTION	1
11	61	Static_Connection_to_DB_5	DBCONNECTION	1
11	60	Static_Connection_to_DB_4	DBCONNECTION	1
11	59	Static_Connection_to_DB_3	DBCONNECTION	1
11	58	Static_Connection_to_DB_2	DBCONNECTION	1
11	57	Static_Connection_to_DB_1	DBCONNECTION	1
11	56	Dynamic_Connection_to_DB	DBCONNECTION	2

Workspace_connections

workspace_i	11
connection_i	17
connection_name	DBCONNECTION
connection_valu	DRIVER=SQL
descriptio	
no_count_displa	0
no_execute	0
parse_query_onl	0
ANSI_quoted_identifier	0
ANSI_nulls	-1
show_query_pla	0
show_stats_tim	0
show_stats_i	0
parse_odbc_msg_prefix	-1
row_count	0
tsql_batch_separat	GO
query_time_out	0
server_languag	(Default)
character_translatio	-1
regional_setting	0

Att_steps

workspace_id = 11

step_label = Setup ACID Tables and Values ***step_id =*** 920 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***

step_file_name = ***version_no =*** 7.0

start_directory = ***parent_version_no =*** 0.0

step_text =

step_label = TPC-H Atomicity ***step_id =*** 921 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = TPC-H Isolation ***step_id =*** 922 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = TPC-H Consistency ***step_id =*** 923 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = TPC-H Durability (Pre-Test Components) ***step_id =*** 924 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 7.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = TPC-H Durability (Post-Test Components) ***step_id =*** 925 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 5.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cleanup ACID Tables and Stored Procs ***step_id =*** 926 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Create ACID History Tables ***step_id =*** 927 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 920 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\acidproc\acid_hist.sql ***version_no =*** 0.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 7.0
step_text =

step_label = Create ACID Values Table ***step_id =*** 928 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 920 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\acidproc\acid_table.sql ***version_no =*** 0.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 7.0
step_text =

step_label = Create ACID Stored Procedures ***step_id =*** 929 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 920 ***enabled_flag =*** -1
iterator_name = PROC_NAME ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = %ACID_DIR% ***parent_version_no =*** 7.0
step_text = osql -Usa -P -S%SERVER_NAME% -d%DBNAME% -iacidproc\%PROC_NAME%

step_label = Populate ACID Values Table ***step_id =*** 930 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 1 ***parent_step_id =*** 920 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = %ACID_DIR% ***parent_version_no =*** 7.0
step_text = osql -Usa -P -S%SERVER_NAME% -d%DBNAME% -w 128 -Q"tpch_acid_values %NUM_STREAMS%"

step_label = Execute Atomicity Commit Test ***step_id =*** 931 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 921 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\atom\scripts\atom_c.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 1.0
step_text =

step_label = Execute Atomicity Rollback Test ***step_id =*** 932 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 921 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\atom\scripts\atom_r.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 1.0
step_text =

step_label = Copy Output Files to Atomicity Directory ***step_id =*** 933 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 921 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 1.0
step_text =

```
::
:: Delete any old files first
::
IF EXIST %ACID_DIR%\atom\atomc.ver DEL %ACID_DIR%\atom\atomc.ver > nul
IF EXIST %ACID_DIR%\atom\atomrb.ver DEL %ACID_DIR%\atom\atomrb.ver > nul

::
:: Now copy the output from the RUN_ID directory
::
copy %OUTPUT_DIR%\Execute_Atomicity_Commit_Test.out %ACID_DIR%\atom\atomc.ver
copy %OUTPUT_DIR%\Execute_Atomicity_Rollback_Test.out %ACID_DIR%\atom\atomrb.ver
```

step_label = Clear any Existing Semaphores ***step_id =*** 934 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 922 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 1.0
step_text = %TOOLS_DIR%\Utility\kill semaphore.exe

step_label = Isolation 1 ***step_id =*** 935 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 922 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 3
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Isolation 2 ***step_id =*** 936 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 922 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 3
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Isolation 3 ***step_id =*** 937 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 1 ***parent_step_id =*** 922 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 3
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Isolation 4 ***step_id =*** 938 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 1 ***parent_step_id =*** 922 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 3
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Isolation 5 ***step_id =*** 939 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 1 ***parent_step_id =*** 922 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 3
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Isolation 6 ***step_id =*** 940 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 1 ***parent_step_id =*** 922 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 4
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Clean-Up Consistency Directory ***step_id =*** 941 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 923 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 1.0
step_text =

```
::
:: Delete any existing output files for the consistency streams
::
IF EXIST CONS\Stream%COUNTER%.OUT DEL CONS\Stream%COUNTER%.out > nul
```

step_label = Setup Consistency Directory ***step_id =*** 942 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 923 ***enabled_flag =*** -1

iterator_name = STREAM ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 1.0

step_text =

```
::
:: Build header for each consistency stream output file
::
ECHO CONSISTENCY STREAM %STREAM% OUTPUT > CONS\Stream%STREAM%.out
ECHO ---------------------------- >> CONS\Stream%STREAM%.out
```

step_label = Truncate HISTORY_ACI Table ***step_id =*** 943 ***global_flag =*** 0

sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 923 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 1.0

step_text =

```
--
-- This worker will clear the HISTORY_ACI table used during the durability process
--
TRUNCATE TABLE HISTORY_ACI
```

step_label = Execute Consistency Test 1 ***step_id =*** 944 ***global_flag =*** 0

sequence_no = 4 ***step_level =*** 1 ***parent_step_id =*** 923 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = %ACID_DIR%\cons\scripts\con1.sql ***version_no =*** 0.0

start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 1.0

step_text =

step_label = ACID Transaction Streams ***step_id =*** 945 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 1 ***parent_step_id =*** 923 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %NUM_STREAMS%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Execute Consistency Test 2 ***step_id =*** 946 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 1 ***parent_step_id =*** 923 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\cons\scripts\con2.sql ***version_no =*** 0.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 1.0
step_text =

step_label = Post-Process Consistency Output ***step_id =*** 947 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 1 ***parent_step_id =*** 923 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Clean-Up Durability Directory ***step_id =*** 948 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 924 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 7.0
step_text =

```
::
:: Delete any existing output files for the consistency streams
::
IF EXIST DUR\Stream%COUNTER%.OUT DEL DURS\Stream%COUNTER%.out > nul
```

step_label = Setup Durability Directory ***step_id =*** 949 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 924 ***enabled_flag =*** -1
iterator_name = STREAM ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 7.0
step_text =

```
::
:: Build header for each durability stream output file
::
ECHO DURABILITY STREAM %STREAM% OUTPUT > DUR\Stream%STREAM%.out
ECHO -------------------------- >> DUR\Stream%STREAM%.out
```

step_label = Truncate HISTORY_DUR Table ***step_id =*** 950 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 924 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 7.0
step_text =

```
--
-- This worker will clear the HISTORY_DUR table used during the durability process
--
SELECT CONVERT(CHAR,GETDATE(),120)
TRUNCATE TABLE HISTORY_DUR
SELECT COUNT(*) FROM HISTORY_DUR
```

step_label = Execute Durability Verification Test 1 ***step_id =*** 951 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 1 ***parent_step_id =*** 924 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\dur\scripts\dur_verify1.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 7.0
step_text =

step_label = Durability Streams ***step_id =*** 952 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 1 ***parent_step_id =*** 924 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %NUM_STREAMS%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 7.0
step_text =

step_label = Execute Durability Verification Test 2 ***step_id =*** 953 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 925 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\dur\scripts\dur_verify2.sql ***version_no =*** 0.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 5.0
step_text =

step_label = Post Process Durability Output ***step_id =*** 954 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 925 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 5.0
step_text =

step_label = Drop ACID History Table ***step_id =*** 955 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 926 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 0.0

step_text =
```
if exists ( select name from sysobjects where name = 'HISTORY_ACI' )
    drop table HISTORY_ACI
GO

if exists ( select name from sysobjects where name = 'HISTORY_DUR' )
    drop table HISTORY_DUR
GO
```

step_label = Drop ACID Values Table ***step_id =*** 956 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 926 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 0.0

step_text =
```
if exists ( select name from sysobjects where name = 'acid_values' )
    drop table acid_values
```

step_label = Drop ACID Stored Procedures ***step_id =*** 957 ***global_flag =*** 0

sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 926 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 0.0

step_text =

```
if exists ( select name from sysobjects where name = 'acid_tran_com' )
      drop procedure acid_tran_com
GO
if exists ( select name from sysobjects where name = 'acid_dur_tran_com' )
      drop procedure acid_dur_tran_com
GO
if exists ( select name from sysobjects where name = 'tran_w_com' )
      drop procedure tran_w_com
GO
if exists ( select name from sysobjects where name = 'tran_w_rb' )
      drop procedure tran_w_rb
GO
if exists ( select name from sysobjects where name = 'iso3_ver' )
      drop procedure iso3_ver
GO
if exists ( select name from sysobjects where name = 'iso5_parts' )
      drop procedure iso5_parts
GO
if exists ( select name from sysobjects where name = 'iso_query' )
      drop procedure iso_query
GO
if exists ( select name from sysobjects where name = 'iso6_tran_w_com' )
      drop procedure iso6_tran_w_com
GO
```

step_label = Prepare Isolation 1 Output ***step_id =*** 958 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 935 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = %ACID_DIR%\iso\scripts\iso1_out.cmd ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text =

step_label = Isolation 1_1 ***step_id =*** 959 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 935 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 1_2 ***step_id =*** 960 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 935 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Prepare Isolation 2 Output ***step_id =*** 961 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 936 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\iso2_out.cmd ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 2_1 ***step_id =*** 962 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 936 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 2_2 ***step_id =*** 963 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 936 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Prepare Isolation 3 Output ***step_id =*** 964 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 937 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\iso3_out.cmd ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 3_1 ***step_id =*** 965 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 937 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 3_2 ***step_id =*** 966 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 937 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Prepare Isolation 4 Output ***step_id =*** 967 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 938 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\iso4_out.cmd ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 4_1 ***step_id =*** 968 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 938 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 4_2 ***step_id =*** 969 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 938 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Prepare Isolation 5 Output ***step_id =*** 970 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 939 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\iso5_out.cmd ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 5_1 ***step_id =*** 971 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 939 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 5_2 ***step_id =*** 972 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 939 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Prepare Isolation 6 Output ***step_id =*** 973 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 940 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\iso6_out.cmd ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 6_1 ***step_id =*** 974 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 940 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 6_2 ***step_id =*** 975 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 940 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Isolation 6_3 ***step_id =*** 976 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 940 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 1 ***step_id =*** 977 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 2 ***step_id =*** 978 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 3 ***step_id =*** 979 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 4 ***step_id =*** 980 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 5 ***step_id =*** 981 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 6 ***step_id =*** 982 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 7 ***step_id =*** 983 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 8 ***step_id =*** 984 ***global_flag =*** 0
sequence_no = 8 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 9 ***step_id =*** 985 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Consistency Stream 10 ***step_id =*** 986 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 2 ***parent_step_id =*** 945 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Durability Stream 1 ***step_id =*** 987 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 952 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Durability Stream 2 ***step_id =*** 988 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 952 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Durability Stream 3 ***step_id =*** 989 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 952 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Durability Stream 4 ***step_id =*** 990 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 952 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Durability Stream 5 *step_id =* 991 *global_flag =* 0
sequence_no = 5 *step_level =* 2 *parent_step_id =* 952 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 0.0
start_directory = *parent_version_no =* 0.0
step_text =

step_label = Durability Stream 6 *step_id =* 992 *global_flag =* 0
sequence_no = 6 *step_level =* 2 *parent_step_id =* 952 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 0.0
start_directory = *parent_version_no =* 0.0
step_text =

step_label = Durability Stream 7 *step_id =* 993 *global_flag =* 0
sequence_no = 7 *step_level =* 2 *parent_step_id =* 952 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 0.0
start_directory = *parent_version_no =* 0.0
step_text =

step_label = Durability Stream 8 *step_id =* 994 *global_flag =* 0
sequence_no = 8 *step_level =* 2 *parent_step_id =* 952 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 0.0
start_directory = *parent_version_no =* 0.0
step_text =

step_label = Durability Stream 9 ***step_id =*** 995 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 2 ***parent_step_id =*** 952 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Durability Stream 10 ***step_id =*** 996 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 2 ***parent_step_id =*** 952 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Prepare .VER Durability Output Files ***step_id =*** 997 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 954 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text =

```
::
:: Copy and rename the output files from the 2 Consistency tests
:;

COPY %DEFAULT_DIR%\%DUR_RUN_ID%\Execute_Durability_Verification_Test_1.out
DUR\DUR1.Ver
COPY %OUTPUT_DIR%\Execute_Durability_Verification_Test_2.out DUR\DUR2.Ver
```

step_label = Dur - Concatenate Stream Output Files ***step_id =*** 998 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 954 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %NUM_STREAMS%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Prepare .VER Consistency Output Files ***step_id =*** 999 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 947 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text =

```
::
:: Copy and rename the output files from the 2 Consistency tests
:;

COPY %OUTPUT_DIR%\Execute_Consistency_Test_1.out CONS\CONS1.Ver
COPY %OUTPUT_DIR%\Execute_Consistency_Test_2.out CONS\CONS2.Ver
```

step_label = Cons - Concatenate Stream Output Files ***step_id =*** 1000 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 947 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %NUM_STREAMS%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Execute Isolation 1_1 ***step_id =*** 1001 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 959 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol1_1.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL1_1 ***step_id =*** 1002 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 959 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO1_1

step_label = Execute Isolation 1_2 ***step_id =*** 1003 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 960 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol1_2.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_2 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL1_2 ***step_id =*** 1004 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 960 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO1_2

step_label = Execute Isolation 2_1 ***step_id =*** 1005 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 962 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol2_1.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL2_1 ***step_id =*** 1006 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 962 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO2_1

step_label = Execute Isolation 2_2 ***step_id =*** 1007 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 963 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol2_2.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_2 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL2_2 ***step_id =*** 1008 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 963 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO2_2

step_label = Execute Isolation 3_1 ***step_id =*** 1009 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 965 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol3_1.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL3_1 ***step_id =*** 1010 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 965 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO3_1

step_label = Execute Isolation 3_2 ***step_id =*** 1011 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 966 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol3_2.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_2 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL3_2 ***step_id =*** 1012 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 966 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO3_2

step_label = Execute Isolation 4_1 ***step_id =*** 1013 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 968 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol4_1.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL4_1 ***step_id =*** 1014 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 968 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO4_1

step_label = Execute Isolation 4_2 ***step_id =*** 1015 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 969 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol4_2.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_2 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL4_2 ***step_id =*** 1016 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 969 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO4_2

step_label = Execute Isolation 5_1 ***step_id =*** 1017 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 971 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol5_1.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL5_1 ***step_id =*** 1018 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 971 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO5_1

step_label = Execute Isolation 5_2 ***step_id =*** 1019 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 972 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol5_2.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_2 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL5_2 ***step_id =*** 1020 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 972 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO5_2

step_label = Execute Isolation 6_1 ***step_id =*** 1021 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 974 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol6_1.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL6_1 ***step_id =*** 1022 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 974 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO6_1

step_label = Execute Isolation 6_2 ***step_id =*** 1023 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 975 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol6_2.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_2 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL6_2 ***step_id =*** 1024 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 975 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO6_2

step_label = Execute Isolation 6_3 ***step_id =*** 1025 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 976 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %ACID_DIR%\iso\scripts\isol6_3.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_3 ***parent_version_no =*** 0.0
step_text =

step_label = Signal End of ISOL6_3 ***step_id =*** 1026 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 976 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal ISO6_3

step_label = Cons - ACID Transactions Stream 1 ***step_id =*** 1027 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 977 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 2 ***step_id =*** 1028 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 978 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_2 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 3 ***step_id =*** 1029 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 979 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_3 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 4 ***step_id =*** 1030 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 980 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_4 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 5 ***step_id =*** 1031 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 981 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_5 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 6 ***step_id =*** 1032 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 982 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_6 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 7 ***step_id =*** 1033 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 983 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_7 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 8 ***step_id =*** 1034 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 984 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_8 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 9 ***step_id =*** 1035 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 985 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_9 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Cons - ACID Transactions Stream 10 ***step_id =*** 1036 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 986 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_10 ***parent_version_no =*** 0.0
step_text = EXEC cons_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 1 ***step_id =*** 1037 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 987 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_1 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 2 ***step_id =*** 1038 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 988 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_2 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 3 ***step_id =*** 1039 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 989 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_3 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 4 ***step_id =*** 1040 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 990 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_4 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 5 ***step_id =*** 1041 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 991 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_5 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 6 ***step_id =*** 1042 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 992 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_6 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 7 ***step_id =*** 1043 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 993 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_7 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 8 ***step_id =*** 1044 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 994 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_8 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 9 ***step_id =*** 1045 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 995 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_9 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - ACID Transactions Stream 10 ***step_id =*** 1046 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 996 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Static_Connection_to_DB_10 ***parent_version_no =*** 0.0
step_text = EXEC dur_tran %COUNTER%

step_label = Dur - Stream 1 Output ***step_id =*** 1047 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 2 Output ***step_id =*** 1048 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 3 Output ***step_id =*** 1049 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 4 Output ***step_id =*** 1050 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 5 Output ***step_id =*** 1051 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 6 Output ***step_id =*** 1052 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 7 Output ***step_id =*** 1053 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 8 Output ***step_id =*** 1054 ***global_flag =*** 0
sequence_no = 8 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 9 Output ***step_id =*** 1055 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Dur - Stream 10 Output ***step_id =*** 1056 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 3 ***parent_step_id =*** 998 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 1 Output ***step_id =*** 1057 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 2 Output ***step_id =*** 1058 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 3 Output ***step_id =*** 1059 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 4 Output ***step_id =*** 1060 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 5 Output ***step_id =*** 1061 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 6 Output ***step_id =*** 1062 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 7 Output ***step_id =*** 1063 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 8 Output ***step_id =*** 1064 ***global_flag =*** 0
sequence_no = 8 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 9 Output ***step_id =*** 1065 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Stream 10 Output ***step_id =*** 1066 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 3 ***parent_step_id =*** 1000 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Cons - Concatenate Stream 1 Output Files ***step_id =*** 1067 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1057 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text = IF EXIST CONS\Stream1.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_1.%COUNTER%.out >> CONS\Stream1.out

step_label = Cons - Concatenate Stream 2 Output Files ***step_id =*** 1068 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1058 ***enabled_flag =*** -1
iterator_name = COUNTER ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %ACID_DIR% ***parent_version_no =*** 0.0
step_text = IF EXIST CONS\Stream2.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_2.%COUNTER%.out >> CONS\Stream2.out

step_label = Cons - Concatenate Stream 3 Output Files ***step_id =*** 1069 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1059 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST CONS\Stream3.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_3.%COUNTER%.out >> CONS\Stream3.out

step_label = Cons - Concatenate Stream 4 Output Files ***step_id =*** 1070 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1060 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST CONS\Stream4.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_4.%COUNTER%.out >> CONS\Stream4.out

step_label = Cons - Concatenate Stream 5 Output Files ***step_id =*** 1071 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1061 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST CONS\Stream5.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_5.%COUNTER%.out >> CONS\Stream5.out

step_label = Cons - Concatenate Stream 6 Output Files ***step_id =*** 1072 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1062 ***enabled_flag =*** 0

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST CONS\Stream6.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_6.%COUNTER%.out >> CONS\Stream6.out

step_label = Cons - Concatenate Stream 7 Output Files ***step_id =*** 1073 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1063 ***enabled_flag =*** 0

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST CONS\Stream7.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_7.%COUNTER%.out >> CONS\Stream7.out

step_label = Cons - Concatenate Stream 8 Output Files ***step_id =*** 1074 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1064 ***enabled_flag =*** 0

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST CONS\Stream8.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_8.%COUNTER%.out >> CONS\Stream8.out

step_label = Cons - Concatenate Stream 9 Output Files ***step_id =*** 1075 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1065 ***enabled_flag =*** 0

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST CONS\Stream9.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_9.%COUNTER%.out >> CONS\Stream9.out

step_label = Cons - Concatenate Stream 10 Output Files ***step_id =*** 1076 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1066 ***enabled_flag =*** 0

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST CONS\Stream10.out TYPE
%OUTPUT_DIR%\Cons_-_ACID_Transactions_Stream_10.%COUNTER%.out >> CONS\Stream10.out

step_label = Dur - Concatenate Stream 1 Output Files ***step_id =*** 1077 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1047 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST DUR\Stream1.out IF NOT EXIST
%DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_1.%COUNTER%.err TYPE
%DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_1.%COUNTER%.out >>
DUR\Stream1.out

step_label = Dur - Concatenate Stream 2 Output Files *step_id* = 1078 *global_flag* = 0

sequence_no = 1 *step_level* = 4 *parent_step_id* = 1048 *enabled_flag* = -1

iterator_name = COUNTER *degree_parallelism* 1

execution_mechanism = 2 *continuation_criteria* = 2 *failure_details* =

step_file_name = *version_no* = 0.0

start_directory = %ACID_DIR% *parent_version_no* = 0.0

step_text = IF EXIST DUR\Stream2.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_2.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_2.%COUNTER%.out >> DUR\Stream2.out

step_label = Dur - Concatenate Stream 3 Output Files *step_id* = 1079 *global_flag* = 0

sequence_no = 1 *step_level* = 4 *parent_step_id* = 1049 *enabled_flag* = -1

iterator_name = COUNTER *degree_parallelism* 1

execution_mechanism = 2 *continuation_criteria* = 2 *failure_details* =

step_file_name = *version_no* = 0.0

start_directory = %ACID_DIR% *parent_version_no* = 0.0

step_text = IF EXIST DUR\Stream3.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_3.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_3.%COUNTER%.out >> DUR\Stream3.out

step_label = Dur - Concatenate Stream 4 Output Files *step_id* = 1080 *global_flag* = 0

sequence_no = 1 *step_level* = 4 *parent_step_id* = 1050 *enabled_flag* = -1

iterator_name = COUNTER *degree_parallelism* 1

execution_mechanism = 2 *continuation_criteria* = 2 *failure_details* =

step_file_name = *version_no* = 0.0

start_directory = %ACID_DIR% *parent_version_no* = 0.0

step_text = IF EXIST DUR\Stream4.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_4.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_4.%COUNTER%.out >> DUR\Stream4.out

step_label = Dur - Concatenate Stream 5 Output Files ***step_id =*** 1081 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1051 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST DUR\Stream5.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_5.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_5.%COUNTER%.out >> DUR\Stream5.out

step_label = Dur - Concatenate Stream 6 Output Files ***step_id =*** 1082 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1052 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST DUR\Stream6.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_6.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_6.%COUNTER%.out >> DUR\Stream6.out

step_label = Dur - Concatenate Stream 7 Output Files ***step_id =*** 1083 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1053 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST DUR\Stream7.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_7.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_7.%COUNTER%.out >> DUR\Stream7.out

step_label = Dur - Concatenate Stream 8 Output Files ***step_id =*** 1084 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1054 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST DUR\Stream8.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_8.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_8.%COUNTER%.out >> DUR\Stream8.out

step_label = Dur - Concatenate Stream 9 Output Files ***step_id =*** 1085 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1055 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST DUR\Stream9.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_9.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_9.%COUNTER%.out >> DUR\Stream9.out

step_label = Dur - Concatenate Stream 10 Output Files ***step_id =*** 1086 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1056 ***enabled_flag =*** -1

iterator_name = COUNTER ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %ACID_DIR% ***parent_version_no =*** 0.0

step_text = IF EXIST DUR\Stream10.out IF NOT EXIST %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_10.%COUNTER%.err TYPE %DEFAULT_DIR%\%DUR_RUN_ID%\Dur_-_ACID_Transactions_Stream_10.%COUNTER%.out >> DUR\Stream10.out

Iterator_values

workspace_id	*step_id*	*version_no*	*type*	*iterator_value*	*sequence_no*
11	1040	0.0	2	810	0
11	1043	0.0	1	1211	0
11	1042	0.0	3	1	0
11	1042	0.0	1	1011	0
11	1042	0.0	2	1210	0
11	1041	0.0	3	1	0
11	1041	0.0	2	1010	0
11	941	0.0	1	1	0
11	1040	0.0	3	1	0
11	1044	0.0	2	1610	0
11	1040	0.0	1	611	0
11	1039	0.0	2	610	0
11	1039	0.0	3	1	0
11	1039	0.0	1	411	0
11	1038	0.0	1	211	0
11	1038	0.0	2	410	0
11	1041	0.0	1	811	0
11	1046	0.0	2	2010	0
11	949	0.0	2	%NUM_STREAMS%	0
11	948	0.0	1	1	0
11	948	0.0	2	10	0
11	948	0.0	3	1	0

11	942	0.0	1	1		0
11	942	0.0	2	%NUM_STREAMS%		0
11	1043	0.0	2	1410		0
11	1046	0.0	3	1		0
11	1043	0.0	3	1		0
11	1046	0.0	1	1811		0
11	1045	0.0	3	1		0
11	1045	0.0	2	1810		0
11	1045	0.0	1	1611		0
11	1044	0.0	1	1411		0
11	1044	0.0	3	1		0
11	1037	0.0	2	210		0
11	942	0.0	3	1		0
11	1028	0.0	3	1		0
11	1038	0.0	3	1		0
11	1031	0.0	1	201		0
11	1030	0.0	3	1		0
11	1030	0.0	1	151		0
11	1030	0.0	2	250		0
11	1029	0.0	3	1		0
11	1031	0.0	2	300		0
11	1029	0.0	1	101		0
11	1032	0.0	1	251		0
11	1028	0.0	1	51		0
11	1028	0.0	2	150		0
11	1027	0.0	3	1		0
11	1027	0.0	2	110		0
11	1027	0.0	1	11		0
11	941	0.0	2	10		0
11	941	0.0	3	1		0

11	1029	0.0	2	200	0
11	1034	0.0	1	351	0
11	1067	0.0	1	11	0
11	1037	0.0	1	11	0
11	1036	0.0	1	451	0
11	1036	0.0	3	1	0
11	1036	0.0	2	550	0
11	1035	0.0	3	1	0
11	1031	0.0	3	1	0
11	1035	0.0	1	401	0
11	1037	0.0	3	1	0
11	1034	0.0	2	450	0
11	1034	0.0	3	1	0
11	1033	0.0	1	301	0
11	1033	0.0	2	400	0
11	1033	0.0	3	1	0
11	1032	0.0	3	1	0
11	1032	0.0	2	350	0
11	1035	0.0	2	500	0
11	1081	0.0	1	811	0
11	1078	0.0	3	1	0
11	1083	0.0	1	1211	0
11	1083	0.0	2	1410	0
11	1083	0.0	3	1	0
11	1082	0.0	3	1	0
11	1082	0.0	2	1210	0
11	1084	0.0	1	1411	0
11	1081	0.0	3	1	0
11	1084	0.0	2	1610	0
11	1081	0.0	2	1010	0

11	1080	0.0	2	810	0
11	1080	0.0	3	1	0
11	1080	0.0	1	611	0
11	1079	0.0	3	1	0
11	1079	0.0	2	610	0
11	949	0.0	1	1	0
11	1082	0.0	1	1011	0
11	929	1.0	4	acid_dur_tran.sql	6
11	929	1.0	4	dur_tran.sql	8
11	929	1.0	4	iso34_tran_w_com.sql	10
11	929	1.0	4	acid_tran.sql	1
11	929	1.0	4	tran_w_com.sql	2
11	929	1.0	4	acid_values.sql	0
11	929	1.0	4	tran_w_rb.sql	3
11	1084	0.0	3	1	0
11	929	1.0	4	iso5_parts.sql	5
11	1078	0.0	2	410	0
11	929	1.0	4	iso6_tran_w_com.sql	9
11	1086	0.0	1	1811	0
11	1086	0.0	3	1	0
11	1086	0.0	2	2010	0
11	1085	0.0	3	1	0
11	1085	0.0	2	1810	0
11	1085	0.0	1	1611	0
11	929	1.0	4	iso_query.sql	4
11	1069	0.0	2	200	0
11	1079	0.0	1	411	0
11	1071	0.0	3	1	0
11	1071	0.0	1	201	0
11	1071	0.0	2	300	0

11	1070	0.0	1	151		0
11	1070	0.0	2	250		0
11	1072	0.0	3	1		0
11	1069	0.0	1	101		0
11	1072	0.0	2	350		0
11	1069	0.0	3	1		0
11	1068	0.0	1	51		0
11	1068	0.0	2	150		0
11	1068	0.0	3	1		0
11	1067	0.0	3	1		0
11	1067	0.0	2	110		0
11	929	1.0	4	cons_tran.sql		7
11	1070	0.0	3	1		0
11	1075	0.0	2	500		0
11	1078	0.0	1	211		0
11	1077	0.0	2	210		0
11	1077	0.0	1	11		0
11	1077	0.0	3	1		0
11	1076	0.0	3	1		0
11	1076	0.0	1	451		0
11	1072	0.0	1	251		0
11	1075	0.0	1	401		0
11	949	0.0	3	1		0
11	1075	0.0	3	1		0
11	1074	0.0	3	1		0
11	1074	0.0	2	450		0
11	1074	0.0	1	351		0
11	1073	0.0	2	400		0
11	1073	0.0	1	301		0
11	1073	0.0	3	1		0

11	1076	0.0	2	550	0

Workspace_parameters

workspace_id =	12	***parameter_id***	202	***parameter_type =***	0
parameter_name	FLATFILE_DIR_16				
parameter_valu	Z:\dev\b016\flat_files_16				
workspace_id =	12	***parameter_id***	204	***parameter_type =***	0
parameter_name	BCP_ROWS				
parameter_valu	10000				
workspace_id =	12	***parameter_id***	203	***parameter_type =***	0
parameter_name	FLAT_FILE_PARALLELISM				
parameter_valu	8				
workspace_id =	12	***parameter_id***	195	***parameter_type =***	0
parameter_name	FLATFILE_DIR_9				
parameter_valu	Z:\dev\b009\flat_files_9				
workspace_id =	12	***parameter_id***	196	***parameter_type =***	0
parameter_name	FLATFILE_DIR_10				
parameter_valu	Z:\dev\b010\flat_files_10				
workspace_id =	12	***parameter_id***	197	***parameter_type =***	0
parameter_name	FLATFILE_DIR_11				
parameter_valu	Z:\dev\b011\flat_files_11				
workspace_id =	12	***parameter_id***	198	***parameter_type =***	0
parameter_name	FLATFILE_DIR_12				
parameter_valu	Z:\dev\b012\flat_files_12				
workspace_id =	12	***parameter_id***	199	***parameter_type =***	0
parameter_name	FLATFILE_DIR_13				
parameter_valu	Z:\dev\b013\flat_files_13				
workspace_id =	12	***parameter_id***	168	***parameter_type =***	0
parameter_name	FLATFILE_DIR_1				
parameter_valu	Z:\dev\b001\flat_files_1				

workspace_id = 12 *parameter_id* 201 *parameter_type* = 0
parameter_name FLATFILE_DIR_15
parameter_valu Z:\dev\b015\flat_files_15

workspace_id = 12 *parameter_id* 147 *parameter_type* = 0
parameter_name SETUP_DIR
parameter_valu %KIT_DIR%\Setup

workspace_id = 12 *parameter_id* 191 *parameter_type* = 0
parameter_name FLATFILE_DIR_5
parameter_valu Z:\dev\b005\flat_files_5

workspace_id = 12 *parameter_id* 192 *parameter_type* = 0
parameter_name FLATFILE_DIR_6
parameter_valu Z:\dev\b006\flat_files_6

workspace_id = 12 *parameter_id* 193 *parameter_type* = 0
parameter_name FLATFILE_DIR_7
parameter_valu Z:\dev\b007\flat_files_7

workspace_id = 12 *parameter_id* 194 *parameter_type* = 0
parameter_name FLATFILE_DIR_8
parameter_valu Z:\dev\b008\flat_files_8

workspace_id = 12 *parameter_id* 190 *parameter_type* = 0
parameter_name TOTAL_LOAD_FILES_PER_TABLE
parameter_valu 96

workspace_id = 12 *parameter_id* 189 *parameter_type* = 0
parameter_name FILES_PER_UPDATE_SET
parameter_valu 64

workspace_id = 12 *parameter_id* 185 *parameter_type* = 0
parameter_name SERVER
parameter_valu asama

workspace_id = 12 *parameter_id* 200 *parameter_type* = 0
parameter_name FLATFILE_DIR_14
parameter_valu Z:\dev\b014\flat_files_14

workspace_id = 12 *parameter_id* 157 *parameter_type* = 3
parameter_name DEFAULT_DIR
parameter_valu %KIT_DIR%\OUTPUT

workspace_id = 12 *parameter_id* 166 *parameter_type* = 0
parameter_name FLATFILE_DIR_3
parameter_valu Z:\dev\b003\flat_files_3

workspace_id = 12 *parameter_id* 165 *parameter_type* = 0
parameter_name FLATFILE_DIR_4
parameter_valu Z:\dev\b004\flat_files_4

workspace_id = 12 *parameter_id* 164 *parameter_type* = 3
parameter_name RUN_ID
parameter_valu 56

workspace_id = 12 *parameter_id* 163 *parameter_type* = 0
parameter_name KIT_DIR
parameter_valu C:\mstpch.101d

workspace_id = 12 *parameter_id* 162 *parameter_type* = 0
parameter_name RUN_DIR
parameter_valu %KIT_DIR%\run

workspace_id = 12 *parameter_id* 161 *parameter_type* = 0
parameter_name QUERY_DIR
parameter_valu %RUN_DIR%\Queries

workspace_id = 12 *parameter_id* 160 *parameter_type* = 0
parameter_name TEMPLATE_DIR
parameter_valu %RUN_DIR%\templates

workspace_id = 12 *parameter_id* 145 *parameter_type* = 0
parameter_name TRACEFLAGS
parameter_valu -c -x -E -T834 -T2301 -T3502

workspace_id = 12 *parameter_id* 158 *parameter_type* = 0
parameter_name TOOLS_DIR
parameter_valu %KIT_DIR%\tools

workspace_id = 12 *parameter_id* 146 *parameter_type* = 0
parameter_name DBNAME
parameter_valu tpch3t

workspace_id = 12 *parameter_id* 156 *parameter_type* = 0
parameter_name RF_FLATFILE_DIR
parameter_valu Z:\dev\b017\tpch3t_RF_FlatFile_DIR

workspace_id = 12 *parameter_id* 155 *parameter_type* = 0
parameter_name TABLE_LOAD_PARALLELISM
parameter_valu 16

workspace_id = 12 *parameter_id* 154 *parameter_type* = 0
parameter_name INDEX_CREATE_PARALLELISM
parameter_valu 3

workspace_id = 12 *parameter_id* 153 *parameter_type* = 0
parameter_name VALIDATION_DIR
parameter_valu %SETUP_DIR%\Validation

workspace_id = 12 *parameter_id* 150 *parameter_type* = 0
parameter_name UPDATE_SETS
parameter_valu 36

workspace_id = 12 *parameter_id* 149 *parameter_type* = 0
parameter_name SCALEFACTOR
parameter_valu 3000

workspace_id = 12 *parameter_id* 148 *parameter_type* = 3
parameter_name OUTPUT_DIR
parameter_valu C:\MSTPCH~1.101\OUTPUT\56

workspace_id = 12 *parameter_id* 167 *parameter_type* = 0
parameter_name FLATFILE_DIR_2
parameter_valu Z:\dev\b002\flat_files_2

workspace_id = 12 *parameter_id* 159 *parameter_type* = 0
parameter_name MAX_STREAMS
parameter_valu 8

Connection_dtls

workspace_id	connection_name_id	connection_name	connection_string_name	connection_type
12	88	Dynamic_connection_to_MASTER_Node1	MASTERDBCONNECTION_NODE1	2
12	89	Dynamic_connection_to_MASTER_Node2	MASTERDBCONNECTION_NODE2	2
12	90	Dynamic_connection_to_MASTER_Node3	MASTERDBCONNECTION_NODE3	2
12	91	Dynamic_connection_to_MASTER_Node4	MASTERDBCONNECTION_NODE4	2
12	70	Static_Connection_to_Master	MASTERDBCONNECTION	1
12	69	Static_Connection_to_DB	DBCONNECTION	1
12	68	Dynamic_Connection_to_Master	MASTERDBCONNECTION	2
12	67	Dynamic_Connection_to_DB	DBCONNECTION	2

Workspace_connections

workspace_i	12
connection_i	19
connection_name	MASTERDBCONNECTION
connection_valu	DRIVER=SQL Server;SERVER=%SERVER%;UID=sa;PWD=;
descriptio	
no_count_displa	0
no_execute	0
parse_query_onl	0
ANSI_quoted_identifier	0
ANSI_nulls	-1
show_query_pla	0
show_stats_tim	0
show_stats_i	0
parse_odbc_msg_prefix	-1
row_count	0
tsql_batch_separat	GO
query_time_out	1
server_languag	(Default)
character_translatio	-1
regional_setting	0
workspace_i	12

dconnection_i 18
connection_name DBCONNECTION
connection_valu DRIVER=SQL Server;SERVER=%SERVER%;UID=sa;PWD=;DATABASE=%DBNAME%;
descriptio
no_count_displa 0
no_execute 0
parse_query_onl 0
ANSI_quoted_identifier 0
ANSI_nulls -1
show_query_pla 0
show_stats_tim 0
show_stats_i 0
parse_odbc_msg_prefix -1
row_count 0
tsql_batch_separat GO
query_time_out 1
server_languag (Default)
character_translatio -1
regional_setting 0

Att_steps

workspace_id = 12

step_label = SqlServer Startup *step_id =* 1087 *global_flag =* -1
sequence_no = 1 *step_level =* 0 *parent_step_id =* 0 *enabled_flag =* 0
iterator_name = *degree_parallelism* 0
execution_mechanism = 2 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 4.0
start_directory = *parent_version_no =* 0.0
step_text = start /D "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn" sqlservr %TRACEFLAGS%
%TOOLS_DIR%\Utility\wait4sql 900000

step_label = SqlServer Shutdown ***step_id =*** 1088 ***global_flag =*** -1
sequence_no = 2 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 0
execution_mechanism = 2 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 3.0
start_directory = ***parent_version_no =*** 0.0
step_text = osql -Usa -P -t10 -Q"shutdown"
%TOOLS_DIR%\Utility\wait4sql 10000

step_label = Wait For SQL Server ***step_id =*** 1089 ***global_flag =*** -1
sequence_no = 3 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 0
execution_mechanism = 2 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\wait4sql.exe 60000

step_label = Generate FlatFiles ***step_id =*** 1090 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 23.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Start SqlServer ***step_id =*** 1091 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 15.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Create database ***step_id =*** 1092 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 88.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Configure SqlServer ***step_id =*** 1093 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 58.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Create and Load Tables ***step_id =*** 1094 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 94.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Create Indexes and Statistics ***step_id =*** 1095 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 46.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Prep Database, End of Load ***step_id =*** 1096 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 79.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Backup TPC-H Database (If not used for ACID) ***step_id =*** 1097 ***global_flag =*** 0
sequence_no = 8 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 24.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Capture Database and Table Space Used (Do Not Use on ***step_id =*** 1098 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 6.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Generate Queries via QGen ***step_id =*** 1099 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 17.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Validation (for SCALEFACTOR=1 Only) ***step_id =*** 1100 ***global_flag =*** 0
sequence_no = 11 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Generate Query Plans ***step_id =*** 1101 ***global_flag =*** 0
sequence_no = 12 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 14.0
start_directory = ***parent_version_no =*** 0.0
step_text =

step_label = Generate FlatFiles in Parallel ***step_id =*** 1102 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1090 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** %TOTAL_LOAD_FILES_PER_TABLE%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 10.0
start_directory = ***parent_version_no =*** 23.0
step_text =

step_label = Generate Update Files ***step_id =*** 1103 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1090 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 4.0

start_directory = %RF_FLATFILE_DIR% ***parent_version_no =*** 23.0

step_text = %TOOLS_DIR%\DBGEN\dbgen -q -b %TOOLS_DIR%\dists.dss -U %UPDATE_SETS% -s %SCALEFACTOR% -f -C %UPDATE_SETS% -i %FILES_PER_UPDATE_SET% -d %FILES_PER_UPDATE_SET%

step_label = Start SqlServer ***step_id =*** 1104 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1091 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 1 ***failure_details =***

step_file_name = ***version_no =*** 4.0

start_directory = C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn ***parent_version_no =*** 15.0

step_text = start /D "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn" sqlservr %TRACEFLAGS% %TOOLS_DIR%\Utility\wait4sql 900000

step_label = Create TPC-H Database ***step_id =*** 1105 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1092 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***

step_file_name = %SETUP_DIR%\%DBNAME%\CreateDatabase.sql ***version_no =*** 18.0

start_directory = Static_Connection_to_Master ***parent_version_no =*** 88.0

step_text =

step_label = Set Database Options ***step_id =*** 1106 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1092 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = Static_Connection_to_Master ***parent_version_no =*** 88.0
step_text =

```
sp_dboption %DBNAME%, 'trunc. log on chkpt.',true
GO
sp_dboption '%DBNAME%','auto create statistics','OFF'
GO
sp_dboption '%DBNAME%','auto update statistics','OFF'
GO
alter database %DBNAME% set PAGE_VERIFY NONE
GO
```

step_label = Move TempDB ***step_id =*** 1108 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1093 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\tempdb\MoveTempDB.sql ***version_no =*** 21.0
start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 58.0
step_text =

step_label = ReSize TempDB ***step_id =*** 1109 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 1093 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\tempdb\ReSizeTempDB.sql ***version_no =*** 19.0
start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 58.0
step_text =

step_label = (Drop/)Create Tables ***step_id =*** 1111 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1094 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 13.0
start_directory = ***parent_version_no =*** 94.0
step_text =

step_label = Parallel Partitioned Table Load ***step_id =*** 1112 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 1094 ***enabled_flag =*** -1
iterator_name = TABLE ***degree_parallelism*** %TABLE_LOAD_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 34.0
start_directory = ***parent_version_no =*** 94.0
step_text =

step_label = Parallel Load Simple Tables ***step_id =*** 1113 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 1 ***parent_step_id =*** 1094 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %TABLE_LOAD_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 13.0
start_directory = ***parent_version_no =*** 94.0
step_text =

step_label = Create NC Indexes in Parallel ***step_id =*** 1116 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 1095 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %INDEX_CREATE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 8.0
start_directory = ***parent_version_no =*** 46.0
step_text =

step_label = Reset DB_Option ***step_id =*** 1119 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1096 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = ***version_no =*** 10.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 79.0
step_text = sp_dboption %DBNAME%, 'trunc. log on chkpt.',false
GO

step_label = Install Refresh Function Stored Procedures ***step_id =*** 1120 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1096 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %MAX_STREAMS%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 7.0
start_directory = ***parent_version_no =*** 79.0
step_text =

step_label = Backup TPC-H Database (Only if using backup to satisfy ***step_id =*** 1121 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 1096 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 3.0
start_directory = ***parent_version_no =*** 79.0
step_text =

step_label = End of Load ***step_id =*** 1122 ***global_flag =*** 0

sequence_no = 4 ***step_level =*** 1 ***parent_step_id =*** 1096 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***

step_file_name = ***version_no =*** 8.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 79.0

step_text =

```
UPDATE TPCH_AUX_TABLE SET LoadFinish = getdate()
GO

SELECT      LoadStart    AS 'Load Start TimeStamp',
     LoadFinish  AS 'Load Finish TimeStamp',
     QgenSeed  AS 'Generated QGen Seed'
FROM    TPCH_AUX_TABLE
GO
```

step_label = Verify TPC-H Load ***step_id =*** 1123 ***global_flag =*** 0

sequence_no = 5 ***step_level =*** 1 ***parent_step_id =*** 1096 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = %SETUP_DIR%\Utility\VerifyTpchLoad.sql ***version_no =*** 4.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 79.0

step_text =

step_label = Generate QGEN Seed ***step_id =*** 1124 ***global_flag =*** 0

sequence_no = 6 ***step_level =*** 1 ***parent_step_id =*** 1096 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 7.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 79.0

step_text =

```
DECLARE @Finish datetime
DECLARE @seed0 integer

--
-- Get ending time of database load
--
SELECT @Finish=LoadFinish FROM TPCH_AUX_TABLE

--
-- Calculate seed per clause 2.1.3.3
--
SET @seed0 =
CONVERT(integer,100000000*DATEPART(MM,@Finish)+1000000*DATEPART(DD,@Finish)+10000*DAT
EPART(HH,@Finish)+100*DATEPART(MI,@Finish)+DATEPART(SS,@Finish))

--
-- Update the benchmark auxillary table
--
UPDATE TPCH_AUX_TABLE SET QgenSeed=@seed0

SELECT * from TPCH_AUX_TABLE
```

step_label = Create Backup Device(s) ***step_id =*** 1125 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1097 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = %SETUP_DIR%\%DBNAME%\CreateBackupDevices.sql ***version_no =*** 3.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 24.0

step_text =

step_label = Execute the backup ***step_id =*** 1126 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1097 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\BackupDatabase.sql ***version_no =*** 3.0
start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 24.0
step_text =

step_label = Execute DBCC UPDATEUSAGE ***step_id =*** 1127 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1098 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 6.0
step_text =

```
--
-- Correct any potential inaccuracies in the system tables before running sp_spaceused
--

dbcc updateusage (%DBNAME%)
GO

dbcc updateusage (tempdb)
GO
```

step_label = Execute SpaceUsed Procedure ***step_id =*** 1128 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1098 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 1.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 6.0

step_text =

```
sp_spaceused
GO
sp_spaceused 'REGION'
GO
sp_spaceused 'NATION'
GO
sp_spaceused 'PART'
GO
sp_spaceused 'SUPPLIER'
GO
sp_spaceused 'PARTSUPP'
GO
sp_spaceused 'CUSTOMER'
GO
sp_spaceused 'ORDERS'
GO
sp_spaceused 'LINEITEM'
GO
```

step_label = Execute Table Row Counts ***step_id =*** 1129 ***global_flag =*** 0

sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 1098 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 1.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 6.0

step_text =

```
print 'Count of REGION Table'
GO
select count(*) from REGION
GO
print 'Count of NATION Table'
GO
select count(*) from NATION
GO
print 'Count of PART Table'
GO
select count(*) from PART
GO
print 'Count of SUPPLIER Table'
GO
select count(*) from SUPPLIER
GO
print 'Count of PARTSUPP Table'
GO
select count_big(*) from PARTSUPP
GO
print 'Count of CUSTOMER Table'
GO
select count(*) from CUSTOMER
GO
print 'Count of ORDERS Table'
GO
select count_big(*) from ORDERS
GO
print 'Count of LINEITEM Table'
GO
select count_big(*) from LINEITEM
GO
```

step_label = Execute Log File Spaceused ***step_id =*** 1130 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 1 ***parent_step_id =*** 1098 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 6.0
step_text = dbcc sqlperf(logspace)

step_label = Execute TempDB spaceused ***step_id =*** 1131 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 1 ***parent_step_id =*** 1098 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 6.0
step_text =
```
use tempdb
GO

sp_spaceused
GO

use %DBNAME%
GO
```

step_label = Generate Power Queries ***step_id =*** 1132 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1099 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 2.2
start_directory = ***parent_version_no =*** 17.0
step_text =

step_label = Generate Stream Queries ***step_id =*** 1133 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 1099 ***enabled_flag =*** -1
iterator_name = STREAM_NUM ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 2.1
start_directory = ***parent_version_no =*** 17.0
step_text =

step_label = Generate Validation Queries via QGEN ***step_id =*** 1134 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1100 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %MAX_STREAMS%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Execute Validation Queries ***step_id =*** 1135 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1100 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 1.0
step_text =

step_label = Relocate Validation Query Output Files ***step_id =*** 1136 ***global_flag =*** 0

sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 1100 ***enabled_flag =*** -1

iterator_name = QUERY ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %VALIDATION_DIR% ***parent_version_no =*** 1.0

step_text =

```
::
:: Copy the StepMaster Query output file to the Validation\Run_Output directory
::
copy %OUTPUT_DIR%\Validation_Query_%QUERY%.out
%VALIDATION_DIR%\RUN_OUTPUT\Validation_Query_%QUERY%.out
```

step_label = Execute SHOWPLAN ALL for Each Query ***step_id =*** 1137 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1101 ***enabled_flag =*** -1

iterator_name = QUERY ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 1.0

start_directory = %SETUP_DIR% ***parent_version_no =*** 14.0

step_text =

```
::
::  First Build the query with the appropriate SQL ShowPlan commands
::
copy %SETUP_DIR%\showplan_sql\set_showplan_all_on.sql
%SETUP_DIR%\showplan_queries\SP_ALL_%QUERY%.SQL
type  %QUERY_DIR%\Power\%QUERY%.SQL                    >>
%SETUP_DIR%\showplan_queries\SP_ALL_%QUERY%.SQL
type  %SETUP_DIR%\showplan_sql\set_showplan_all_off.sql    >>
%SETUP_DIR%\showplan_queries\SP_ALL_%QUERY%.SQL

::
::  Now execute the query, placing the output in %SETUP_DIR%\Showplan_Out
::
osql -l -t600 -Usa -P -w1000 -S%SERVER% -d %DBNAME% -e -i
%SETUP_DIR%\showplan_queries\SP_ALL_%QUERY%.sql -w 4096 -o
%SETUP_DIR%\showplan_Out\SP_ALL_%QUERY%.out
```

step_label = Execute SHOWPLAN TEXT for Each Query ***step_id =*** 1138 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1101 ***enabled_flag =*** -1

iterator_name = QUERY ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 1.0

start_directory = %SETUP_DIR% ***parent_version_no =*** 14.0

step_text =

```
::
::  First Build the query with the appropriate SQL ShowPlan commands
::
copy %SETUP_DIR%\showplan_sql\set_showplan_text_on.sql
%SETUP_DIR%\showplan_queries\SP_TEXT_%QUERY%.SQL
type  %QUERY_DIR%\Power\%QUERY%.SQL                    >>
%SETUP_DIR%\showplan_queries\SP_TEXT_%QUERY%.SQL
type  %SETUP_DIR%\showplan_sql\set_showplan_text_off.sql    >>
%SETUP_DIR%\showplan_queries\SP_TEXT_%QUERY%.SQL

::
::  Now execute the query, placing the output in %SETUP_DIR%\Showplan_Out
::
osql -l -t600 -Usa -P -w1000 -S%SERVER% -d %DBNAME% -e -i
%SETUP_DIR%\showplan_queries\SP_TEXT_%QUERY%.sql -w 4096 -o
%SETUP_DIR%\showplan_Out\SP_TEXT_%QUERY%.out
```

step_label = Generate Flat Files to FlatFile_Dir_1 ***step_id =*** 1140 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 15.0

start_directory = %FLATFILE_DIR_1% ***parent_version_no =*** 10.0

step_text =

```
:: Generate Data for REGION and NATION Tables
%TOOLS_DIR%\DBGen\dbgen -T l -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_2 ***step_id =*** 1141 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 12.0

start_directory = %FLATFILE_DIR_2% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_3 ***step_id =*** 1142 ***global_flag =*** 0

sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 13.0

start_directory = %FLATFILE_DIR_3% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_4 ***step_id =*** 1143 ***global_flag =*** 0

sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 10.0

start_directory = %FLATFILE_DIR_4% ***parent_version_no =*** 10.0

step_text =

```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Create Auxilliary Table ***step_id =*** 1145 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1111 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = %SETUP_DIR%\Utility\CreateBuildTimer.sql ***version_no =*** 8.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 13.0

step_text =

step_label = Create Base Tables ***step_id =*** 1146 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1111 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***

step_file_name = %SETUP_DIR%\%DBNAME%\CreateTables.sql ***version_no =*** 6.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 13.0

step_text =

step_label = Load Nation Table ***step_id =*** 1155 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1113 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***

step_file_name = ***version_no =*** 1.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 13.0

step_text =
```
bulk insert %DBNAME%..NATION
from '%FLATFILE_DIR_1%\nation.tbl'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock)
```

step_label = Load Region Table ***step_id =*** 1156 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1113 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***

step_file_name = ***version_no =*** 1.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 13.0

step_text =
```
bulk insert %DBNAME%..REGION
from '%FLATFILE_DIR_1%\region.tbl'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock)
```

step_label = CreateIndexStream1 ***step_id =*** 1157 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1116 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***

step_file_name = %SETUP_DIR%\%DBNAME%\CreateIndexesStream1.sql ***version_no =*** 6.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 8.0

step_text =

step_label = CreateIndexStream2 ***step_id =*** 1158 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1116 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\CreateIndexesStream2.sql ***version_no =*** 6.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 8.0
step_text =

step_label = CreateIndexStream3 ***step_id =*** 1159 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1116 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\CreateIndexesStream3.sql ***version_no =*** 5.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 8.0
step_text =

step_label = Create Backup Device(s) (For ACID) ***step_id =*** 1164 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1121 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\CreateBackupDevices.sql ***version_no =*** 0.0
start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 3.0
step_text =

step_label = Execute the backup (For ACID) ***step_id =*** 1165 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1121 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\BackupDatabase.sql ***version_no =*** 0.0
start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 3.0
step_text =

step_label = Install RF1 Stored Procedure(s) ***step_id =*** 1166 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1120 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\CreateRF1Proc.sql ***version_no =*** 7.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 7.0
step_text =

step_label = Install RF2 Stored Procedure(s) ***step_id =*** 1167 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1120 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\CreateRF2Proc.sql ***version_no =*** 7.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 7.0
step_text =

step_label = Generate Power Query Directory ***step_id =*** 1168 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1132 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %RUN_DIR% ***parent_version_no =*** 2.2
step_text = if not exist %QUERY_DIR%\Power mkdir %QUERY_DIR%\Power

step_label = Generate QGen Command File ***step_id =*** 1169 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1132 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = ***version_no =*** 2.0
start_directory = %TEMPLATE_DIR% ***parent_version_no =*** 2.2
step_text =

```
::
:: First use OSQL to grab the QgenSeed value from TPCH_AUX_TABLE
::
osql -Usa -P -n -S%SERVER% -d%DBNAME% -w 255 -i%SETUP_DIR%\Generate\GenQGENcmd.sql >
%SETUP_DIR%\Generate\QgenCmd.cmd
```

step_label = Parallel Power Query Generation ***step_id =*** 1170 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1132 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %MAX_STREAMS%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 2.2
step_text =

step_label = Set Seed Value for Stream ***step_id =*** 1171 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1133 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 2.1
step_text =

```
--
--   Increment QGen Seed in TPCH_AUX_TABLE
--
UPDATE   TPCH_AUX_TABLE
SET      QgenSeed = QgenSeed + %STREAM_NUM%
```

step_label = Create Query Stream Directories ***step_id =*** 1172 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1133 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %QUERY_DIR% ***parent_version_no =*** 2.1

step_text =
```
if not exist %QUERY_DIR%\STREAM%STREAM_NUM% mkdir
%QUERY_DIR%\STREAM%STREAM_NUM%
```

step_label = Generate QGen Stream Command File ***step_id =*** 1173 ***global_flag =*** 0

sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1133 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 1.0

start_directory = %TEMPLATE_DIR% ***parent_version_no =*** 2.1

step_text =
```
::
:: First use OSQL to grab the QgenSeed value from TPCH_AUX_TABLE
::
osql -Usa -P -n -S%SERVER% -d%DBNAME% -w 255
-i%SETUP_DIR%\Generate\GenQgenStreamCmd.sql >
%SETUP_DIR%\Generate\QgenStreamCmd.cmd
```

step_label = Parallel Stream Query Generation ***step_id =*** 1174 ***global_flag =*** 0

sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 1133 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** %MAX_STREAMS%

execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = ***parent_version_no =*** 2.1

step_text =

step_label = Re-set QGen Seed Value ***step_id =*** 1175 ***global_flag =*** 0

sequence_no = 5 ***step_level =*** 2 ***parent_step_id =*** 1133 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 2.1

step_text =

```
--
--  Resets the QGen seed kept in the temporary table
--
UPDATE   TPCH_AUX_TABLE
SET      QgenSeed  =  QgenSeed - %STREAM_NUM%
```

step_label = Generate Validation Queries ***step_id =*** 1176 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1134 ***enabled_flag =*** -1

iterator_name = QUERY ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 0.0

start_directory = %TEMPLATE_DIR% ***parent_version_no =*** 0.0

step_text =

```
%TOOLS_DIR%\QGen\qgen -d -b %TOOLS_DIR%\dists.dss  %QUERY% >
%VALIDATION_DIR%\Queries\v%QUERY%.sql
```

step_label = Validation Query 1 ***step_id =*** 1177 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = %VALIDATION_DIR%\Queries\v1.sql ***version_no =*** 0.0

start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0

step_text =

step_label = Validation Query 2 ***step_id =*** 1178 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v2.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 3 ***step_id =*** 1179 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v3.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 4 ***step_id =*** 1180 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v4.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 5 ***step_id =*** 1181 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v5.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 6 ***step_id =*** 1182 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v6.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 7 ***step_id =*** 1183 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v7.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 8 ***step_id =*** 1184 ***global_flag =*** 0
sequence_no = 8 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v8.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 9 ***step_id =*** 1185 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v9.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 10 ***step_id =*** 1186 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v10.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 11 ***step_id =*** 1187 ***global_flag =*** 0
sequence_no = 11 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v11.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 12 ***step_id =*** 1188 ***global_flag =*** 0
sequence_no = 12 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v12.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 13 ***step_id =*** 1189 ***global_flag =*** 0
sequence_no = 13 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v13.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 14 ***step_id =*** 1190 ***global_flag =*** 0
sequence_no = 14 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v14.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 15 ***step_id =*** 1191 ***global_flag =*** 0
sequence_no = 15 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v15.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 16 ***step_id =*** 1192 ***global_flag =*** 0
sequence_no = 16 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v16.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 17 ***step_id =*** 1193 ***global_flag =*** 0
sequence_no = 17 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v17.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 18 ***step_id =*** 1194 ***global_flag =*** 0
sequence_no = 18 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v18.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 19 ***step_id =*** 1195 ***global_flag =*** 0
sequence_no = 19 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v19.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 20 ***step_id =*** 1196 ***global_flag =*** 0
sequence_no = 20 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v20.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 21 ***step_id =*** 1197 ***global_flag =*** 0
sequence_no = 21 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v21.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Validation Query 22 ***step_id =*** 1198 ***global_flag =*** 0
sequence_no = 22 ***step_level =*** 2 ***parent_step_id =*** 1135 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %VALIDATION_DIR%\Queries\v22.sql ***version_no =*** 0.0
start_directory = Static_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Power - Execute Generated QGen Command File ***step_id =*** 1199 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1170 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = %SETUP_DIR%\Generate\QgenCmd.cmd ***version_no =*** 0.0
start_directory = %TEMPLATE_DIR% ***parent_version_no =*** 0.0
step_text =

step_label = Throughput - Execute Generated QGen Command File ***step_id =*** 1200 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1174 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\Generate\QgenStreamCmd.cmd ***version_no =*** 0.0
start_directory = %TEMPLATE_DIR% ***parent_version_no =*** 0.0
step_text =

step_label = Set SEED to AsAmA/9921.2 seed ***step_id =*** 1277 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1099 ***enabled_flag =*** 0

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 3.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 17.0

step_text =

```
-- Update the benchmark auxillary table
-- Use the AsAmA seed for the 9921.2 QphH result (load end at 2005-04-23 04:09:54.130)
-- raid5 seed 605194558
-- raid0 seed 709161519
UPDATE TPCH_AUX_TABLE SET QgenSeed=709161519

SELECT * from TPCH_AUX_TABLE
```

step_label = Set Correlation ***step_id =*** 1343 ***global_flag =*** 0

sequence_no = 5 ***step_level =*** 1 ***parent_step_id =*** 1095 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 5.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 46.0

step_text = alter database %DBNAME% set date_correlation_optimization ON

step_label = Create Statistics ***step_id =*** 1358 ***global_flag =*** 0

sequence_no = 4 ***step_level =*** 1 ***parent_step_id =*** 1095 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 3.0

start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 46.0

step_text = sp_createstats

step_label = Generate Flat Files to FlatFile_Dir_5 ***step_id =*** 1359 ***global_flag =*** 0

sequence_no = 5 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 12.0

start_directory = %FLATFILE_DIR_5% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_6 ***step_id =*** 1360 ***global_flag =*** 0

sequence_no = 6 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 12.0

start_directory = %FLATFILE_DIR_6% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_7 ***step_id =*** 1361 ***global_flag =*** 0

sequence_no = 7 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 12.0

start_directory = %FLATFILE_DIR_7% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_8 ***step_id =*** 1362 ***global_flag =*** 0

sequence_no = 8 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_8% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Load Partitioned Tables from FlatFile_Dir_1 ***step_id =*** 1363 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1388 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 21.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 2.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_1%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_2 ***step_id =*** 1364 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1389 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 21.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 3.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_2%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_3 ***step_id =*** 1365 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1390 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_3%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_4 *step_id* = 1366 *global_flag* = 0
sequence_no = 1 *step_level* = 3 *parent_step_id* = 1391 *enabled_flag* = -1
iterator_name = PARALLEL_PROCESS *degree_parallelism* 6
execution_mechanism = 1 *continuation_criteria* = 2 *failure_details* =
step_file_name = *version_no* = 19.0
start_directory = Dynamic_Connection_to_Master *parent_version_no* = 4.0
step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_4%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_5 *step_id* = 1367 *global_flag* = 0
sequence_no = 1 *step_level* = 3 *parent_step_id* = 1392 *enabled_flag* = -1
iterator_name = PARALLEL_PROCESS *degree_parallelism* 6
execution_mechanism = 1 *continuation_criteria* = 2 *failure_details* =
step_file_name = *version_no* = 19.0
start_directory = Dynamic_Connection_to_Master *parent_version_no* = 4.0
step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_5%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_6 *step_id* = 1368 *global_flag* = 0
sequence_no = 1 *step_level* = 3 *parent_step_id* = 1393 *enabled_flag* = -1
iterator_name = PARALLEL_PROCESS *degree_parallelism* 6
execution_mechanism = 1 *continuation_criteria* = 2 *failure_details* =
step_file_name = *version_no* = 19.0
start_directory = Dynamic_Connection_to_Master *parent_version_no* = 4.0
step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_6%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_7 ***step_id =*** 1369 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1394 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_7%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_8 ***step_id =*** 1370 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1395 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_8%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_9 ***step_id =*** 1371 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1396 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_9%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_10 ***step_id =*** 1372 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1397 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_10%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_11 ***step_id =*** 1373 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1398 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_11%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_12 ***step_id =*** 1374 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1399 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_12%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_13 ***step_id =*** 1375 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1400 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_13%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_14 ***step_id =*** 1376 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1401 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 20.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_14%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_15 ***step_id =*** 1377 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1402 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_15%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Load Partitioned Tables from FlatFile_Dir_16 ***step_id =*** 1378 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1403 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 6

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 19.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 4.0

step_text =
```
bulk insert %DBNAME%..%TABLE%
from '%FLATFILE_DIR_16%\%TABLE%.tbl.%PARALLEL_PROCESS%'
with (FieldTerminator = '|', RowTerminator ='|\n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)
```

step_label = Generate Flat Files to FlatFile_Dir_9 ***step_id =*** 1379 ***global_flag =*** 0

sequence_no = 9 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_9% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_10 ***step_id =*** 1380 ***global_flag =*** 0

sequence_no = 10 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_10% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_11 ***step_id =*** 1381 ***global_flag =*** 0

sequence_no = 11 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_11% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_12 ***step_id =*** 1382 ***global_flag =*** 0

sequence_no = 12 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_12% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_13 ***step_id =*** 1383 ***global_flag =*** 0

sequence_no = 13 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_13% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_14 ***step_id =*** 1384 ***global_flag =*** 0

sequence_no = 14 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_14% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_15 ***step_id =*** 1385 ***global_flag =*** 0

sequence_no = 15 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_15% ***parent_version_no =*** 10.0

step_text =
```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Generate Flat Files to FlatFile_Dir_16 ***step_id =*** 1386 ***global_flag =*** 0

sequence_no = 16 ***step_level =*** 2 ***parent_step_id =*** 1102 ***enabled_flag =*** -1

iterator_name = PARALLEL_PROCESS ***degree_parallelism*** 1

execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 11.0

start_directory = %FLATFILE_DIR_16% ***parent_version_no =*** 10.0

step_text =

```
:: Generate Data for CUSTOMER Table
%TOOLS_DIR%\DBGen\dbgen -T c -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for ORDERS and LINEITEM Tables
%TOOLS_DIR%\DBGen\dbgen -T o -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for PARTS/PARTSUPP Table
%TOOLS_DIR%\DBGen\dbgen -T p -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%

:: Generate Data for SUPPLIER Table
%TOOLS_DIR%\DBGen\dbgen -T s -fF -q -b %TOOLS_DIR%\dists.dss -s%SCALEFACTOR%
-C%TOTAL_LOAD_FILES_PER_TABLE% -S%PARALLEL_PROCESS%
```

step_label = Files 1 ***step_id =*** 1388 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%

execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***

step_file_name = ***version_no =*** 2.0

start_directory = ***parent_version_no =*** 34.0

step_text =

step_label = Files 2 ***step_id =*** 1389 ***global_flag =*** 0

sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%

execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***

step_file_name = ***version_no =*** 3.0

start_directory = ***parent_version_no =*** 34.0

step_text =

step_label = Files 3 ***step_id =*** 1390 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 4 ***step_id =*** 1391 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 5 ***step_id =*** 1392 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 6 ***step_id =*** 1393 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 7 ***step_id =*** 1394 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 8 ***step_id =*** 1395 ***global_flag =*** 0
sequence_no = 8 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 9 ***step_id =*** 1396 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 10 ***step_id =*** 1397 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 11 ***step_id =*** 1398 ***global_flag =*** 0
sequence_no = 11 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 12 ***step_id =*** 1399 ***global_flag =*** 0
sequence_no = 12 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 13 ***step_id =*** 1400 ***global_flag =*** 0
sequence_no = 13 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 14 ***step_id =*** 1401 ***global_flag =*** 0
sequence_no = 14 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 15 ***step_id =*** 1402 ***global_flag =*** 0
sequence_no = 15 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Files 16 ***step_id =*** 1403 ***global_flag =*** 0
sequence_no = 16 ***step_level =*** 2 ***parent_step_id =*** 1112 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FLAT_FILE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 34.0
step_text =

step_label = Set sp_configure Options ***step_id =*** 1439 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1093 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\Utility\conf_tpch.sql ***version_no =*** 17.0
start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 58.0
step_text =

step_label = Create FK Constraints ***step_id =*** 1441 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1095 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\createFK.sql ***version_no =*** 5.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 46.0
step_text =

step_label = Load Start Timestamp ***step_id =*** 1442 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1094 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** 1

execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***

step_file_name = ***version_no =*** 15.0

start_directory = Dynamic_Connection_to_Master ***parent_version_no =*** 94.0

step_text =

```
--    Create temporary table for timing in the Master Database
--
--          This is just temporary until we build the TPCH_AUX_TABLE later
--


--
-- Delete any existing tpch_temp_timer table
--
if exists ( select name from sysobjects where name = 'tpch_temp_timer' )
      drop table tpch_temp_timer

--
-- Create the temporary table
--
create table tpch_temp_timer
(
     load_start_time                              datetime
)

--
-- Store the starting time in the temporary table
--
insert            into tpch_temp_timer values (getdate())
```

step_label = Create Clustered Indexes in Parallel ***step_id =*** 1444 ***global_flag =*** 0

sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1095 ***enabled_flag =*** -1

iterator_name = ***degree_parallelism*** %INDEX_CREATE_PARALLELISM%

execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***

step_file_name = ***version_no =*** 6.0

start_directory = ***parent_version_no =*** 46.0

step_text =

step_label = Create Clustered Indexes1 ***step_id =*** 1445 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1444 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\CreateClusteredIndexes1.sql ***version_no =*** 8.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 6.0
step_text =

step_label = Create Clustered Indexes2 ***step_id =*** 1446 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1444 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = %SETUP_DIR%\%DBNAME%\CreateClusteredIndexes2.sql ***version_no =*** 8.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 6.0
step_text =

Step_constraints

workspace_id	*constraint_id*	*step_id*	*version_no*	*constraint_type*	*global_step_id*	*global_version_no*	*sequence_no*
12	37	1104	4.0	2	1089	0.0	0
12	33	1108	21.0	2	1087	4.0	1
12	35	1108	21.0	2	1088	3.0	0
12	38	1108	21.0	2	1089	0.0	2

Iterator_values

workspace_id	*step_id*	*version_no*	*type*	*iterator_value*	*sequence_no*
12	1140	15.0	3	1	0
12	1361	12.0	3	1	0
12	1360	12.0	1	31	0

12	1360	12.0	3	1	0
12	1360	12.0	2	36	0
12	1359	12.0	3	1	0
12	1359	12.0	2	30	0
12	1359	12.0	1	25	0
12	1142	13.0	3	1	0
12	1142	13.0	1	13	0
12	1142	13.0	2	18	0
12	1141	12.0	3	1	0
12	1141	12.0	2	12	0
12	1386	11.0	1	91	0
12	1140	15.0	2	6	0
12	1143	10.0	3	1	0
12	1140	15.0	1	1	0
12	1112	34.0	4	ORDERS	1
12	1112	34.0	4	SUPPLIER	2
12	1112	34.0	4	PARTSUPP	5
12	1112	34.0	4	CUSTOMER	4
12	1112	34.0	4	PART	3
12	1112	34.0	4	LINEITEM	0
12	1136	0.0	2	22	0
12	1136	0.0	1	1	0
12	1136	0.0	3	1	0
12	1176	0.0	1	1	0
12	1176	0.0	3	1	0
12	1176	0.0	2	22	0
12	1141	12.0	1	7	0
12	1381	11.0	1	61	0
12	1386	11.0	2	96	0
12	1386	11.0	3	1	0

12	1385	11.0	3 1	0
12	1385	11.0	2 90	0
12	1385	11.0	1 85	0
12	1384	11.0	2 84	0
12	1384	11.0	1 79	0
12	1384	11.0	3 1	0
12	1383	11.0	2 78	0
12	1383	11.0	3 1	0
12	1383	11.0	1 73	0
12	1382	11.0	1 67	0
12	1382	11.0	2 72	0
12	1361	12.0	2 42	0
12	1379	11.0	1 49	0
12	1199	0.0	2 22	0
12	1143	10.0	1 19	0
12	1143	10.0	2 24	0
12	1362	11.0	1 43	0
12	1362	11.0	2 48	0
12	1382	11.0	3 1	0
12	1379	11.0	3 1	0
12	1381	11.0	3 1	0
12	1379	11.0	2 54	0
12	1380	11.0	3 1	0
12	1380	11.0	2 60	0
12	1380	11.0	1 55	0
12	1381	11.0	2 66	0
12	1361	12.0	1 37	0
12	1362	11.0	3 1	0
12	1364	21.0	3 1	0
12	1199	0.0	1 1	0

12	1369	19.0	1	37		0
12	1369	19.0	3	1		0
12	1368	19.0	2	36		0
12	1368	19.0	1	31		0
12	1368	19.0	3	1		0
12	1367	19.0	3	1		0
12	1367	19.0	2	30		0
12	1367	19.0	1	25		0
12	1366	19.0	1	19		0
12	1366	19.0	2	24		0
12	1366	19.0	3	1		0
12	1365	19.0	1	13		0
12	1370	19.0	3	1		0
12	1138	1.0	1	1		0
12	1133	2.1	1	1		0
12	1133	2.1	2	%MAX_STREAMS%		0
12	1137	1.0	2	22		0
12	1137	1.0	3	1		0
12	1137	1.0	1	1		0
12	1365	19.0	2	18		0
12	1138	1.0	2	22		0
12	1365	19.0	3	1		0
12	1363	21.0	1	1		0
12	1363	21.0	2	6		0
12	1363	21.0	3	1		0
12	1364	21.0	1	7		0
12	1364	21.0	2	12		0
12	1370	19.0	2	48		0
12	1138	1.0	3	1		0
12	1375	19.0	1	73		0

12	1133	2.1	3 1	0
12	1200	0.0	1 1	0
12	1200	0.0	2 22	0
12	1200	0.0	3 1	0
12	1378	19.0	3 1	0
12	1378	19.0	2 96	0
12	1378	19.0	1 91	0
12	1377	19.0	1 85	0
12	1377	19.0	3 1	0
12	1377	19.0	2 90	0
12	1376	20.0	3 1	0
12	1376	20.0	1 79	0
12	1376	20.0	2 84	0
12	1369	19.0	2 42	0
12	1373	19.0	3 1	0
12	1370	19.0	1 43	0
12	1371	19.0	3 1	0
12	1371	19.0	2 54	0
12	1371	19.0	1 49	0
12	1372	19.0	3 1	0
12	1375	19.0	3 1	0
12	1372	19.0	1 55	0
12	1375	19.0	2 78	0
12	1373	19.0	2 66	0
12	1373	19.0	1 61	0
12	1374	19.0	2 72	0
12	1374	19.0	3 1	0
12	1374	19.0	1 67	0
12	1199	0.0	3 1	0
12	1372	19.0	2 60	0

Workspace_parameters

workspace_id =	13	*parameter_id*	180	*parameter_type* =	0
parameter_name	TOOLS_DIR				
parameter_valu	%KIT_DIR%\Tools				
workspace_id =	13	*parameter_id*	170	*parameter_type* =	3
parameter_name	OUTPUT_DIR				
parameter_valu	C:\MSTPCH~1.101\output\58				
workspace_id =	13	*parameter_id*	171	*parameter_type* =	0
parameter_name	QUERY_DIR				
parameter_valu	%RUN_DIR%\Queries				
workspace_id =	13	*parameter_id*	172	*parameter_type* =	0
parameter_name	DBNAME				
parameter_valu	tpch3t				
workspace_id =	13	*parameter_id*	173	*parameter_type* =	0
parameter_name	DELETE_PARALLELISM				
parameter_valu	64				
workspace_id =	13	*parameter_id*	174	*parameter_type* =	0
parameter_name	INSERT_PARALLELISM				
parameter_valu	64				
workspace_id =	13	*parameter_id*	175	*parameter_type* =	0
parameter_name	DELETE_EXECUTIONS				
parameter_valu	192				
workspace_id =	13	*parameter_id*	176	*parameter_type* =	0
parameter_name	INSERT_EXECUTIONS				
parameter_valu	192				
workspace_id =	13	*parameter_id*	169	*parameter_type* =	0
parameter_name	RUN_DIR				
parameter_valu	%KIT_DIR%\Run				

workspace_id = 13 *parameter_id* 178 *parameter_type* = 3
parameter_name DEFAULT_DIR
parameter_valu %KIT_DIR%\output

workspace_id = 13 *parameter_id* 208 *parameter_type* = 0
parameter_name SERVER
parameter_valu asama

workspace_id = 13 *parameter_id* 181 *parameter_type* = 0
parameter_name BATCH_SIZE
parameter_valu 30

workspace_id = 13 *parameter_id* 182 *parameter_type* = 0
parameter_name KIT_DIR
parameter_valu C:\mstpch.101d

workspace_id = 13 *parameter_id* 183 *parameter_type* = 3
parameter_name RUN_ID
parameter_valu 58

workspace_id = 13 *parameter_id* 184 *parameter_type* = 0
parameter_name TRACEFLAGS
parameter_valu -c -x -E -T834 -T2301 -T3502

workspace_id = 13 *parameter_id* 186 *parameter_type* = 0
parameter_name SETUP_DIR
parameter_valu %KIT_DIR%\Setup

workspace_id = 13 *parameter_id* 206 *parameter_type* = 0
parameter_name RF_FLATFILE_DIR
parameter_valu Z:\dev\b017\tpch3t_RF_FlatFile_DIR

workspace_id = 13 *parameter_id* 207 *parameter_type* = 0
parameter_name FILES_PER_UPDATE_SET
parameter_valu 64

workspace_id = 13 *parameter_id* 177 *parameter_type* = 0
parameter_name MAX_STREAMS
parameter_valu 8

Connection_dtls

workspace_id	connection_name_id	connection_name	connection_string_name	connection_type
13	84	Power_Dynamic_DB_Connection	DBCONNECTION	2
13	87	Dynamic_Connection_to_Master	MASTERCONNECTION	2
13	83	Throughput_Static_Stream_10_Connection	DBCONNECTION	1
13	82	Throughput_Static_Stream_9_Connection	DBCONNECTION	1
13	81	Throughput_Static_Stream_8_Connection	DBCONNECTION	1
13	80	Throughput_Static_Stream_7_Connection	DBCONNECTION	1
13	79	Throughput_Static_Stream_6_Connection	DBCONNECTION	1
13	78	Throughput_Static_Stream_5_Connection	DBCONNECTION	1
13	77	Throughput_Static_Stream_4_Connection	DBCONNECTION	1
13	76	Throughput_Static_Stream_3_Connection	DBCONNECTION	1
13	75	Throughput_Static_Stream_2_Connection	DBCONNECTION	1
13	74	Throughput_Static_Stream_1_Connection	DBCONNECTION	1
13	73	Throughput_RF_Connection	DBCONNECTION	2
13	72	Dynamic_Connection_to_DB	DBCONNECTION	2
13	71	Power_Static_DB_Connection	DBCONNECTION	1

Workspace_connections

workspace_i	13
connection_i	21
connection_name	MASTERCONNECTION
connection_valu	DRIVER=SQL Server;SERVER=%SERVER%;UID=sa;PWD=;
descriptio	
no_count_displa	0
no_execute	0
parse_query_onl	0
ANSI_quoted_identifier	0
ANSI_nulls	-1
show_query_pla	0
show_stats_tim	0
show_stats_i	0
parse_odbc_msg_prefix	-1
row_count	0
tsql_batch_separat	GO
query_time_out	0
server_languag	(Default)
character_translatio	-1
regional_setting	0

workspace_i	13
connection_i	20
connection_name	DBCONNECTION
connection_valu	DRIVER=SQL Server;SERVER=%SERVER%;UID=sa;PWD=;DATABASE=%DBNAME%;
descriptio	
no_count_displa	0
no_execute	0
parse_query_onl	0
ANSI_quoted_identifier	0
ANSI_nulls	-1
show_query_pla	0
show_stats_tim	0
show_stats_i	0
parse_odbc_msg_prefix	-1
row_count	0
tsql_batch_separat	GO

query_time_out 0
d
server_languag (Default)
d
character_translatio -1
regional_setting 0
e
n

y Att_steps

y workspace_id = 13
s
step_label = Clear any Outstanding Semaphores *step_id =* 1201 *global_flag =* 0
sequence_no = *n* 1 *step_level =* 0 *parent_step_id =* 0 *enabled_flag =* -1
e
iterator_name = *degree_parallelism* 1
o
execution_mechanism = *es* 2 *continuation_criteria =* 2 *failure_details =*
step_file_name = *version_no =* 70.0
start_directory = *or* %TOOLS_DIR% *parent_version_no* = 0.0
step_text = ::
:: This step must always be run to insure that a semaphore.exe was not left open by a
:: previous run
::
:: If there are no open semaphore.exe's then the 'KILL' will do nothing.
::
%TOOLS_DIR%\Utility\KILL.EXE SEMAPHORE.EXE

step_label = Execute Power Run *step_id =* 1202 *global_flag =* 0
sequence_no = 2 *step_level =* 0 *parent_step_id =* 0 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 125.12
start_directory = *parent_version_no =* 0.0
step_text =

step_label = Execute Throughput Run *step_id =* 1203 *global_flag =* 0
sequence_no = 3 *step_level =* 0 *parent_step_id =* 0 *enabled_flag =* -1
iterator_name = *degree_parallelism* 2
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 49.3
start_directory = *parent_version_no =* 0.0
step_text =

step_label = Power - Sequential Query Execution *step_id =* 1205 *global_flag =* 0
sequence_no = 2 *step_level =* 1 *parent_step_id =* 1202 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 26.0
start_directory = *parent_version_no =* 125.12
step_text =

step_label = Power - Increment Update Set *step_id =* 1207 *global_flag =* 0
sequence_no = 4 *step_level =* 1 *parent_step_id =* 1202 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 1 *continuation_criteria =* 2 *failure_details =*
step_file_name = *version_no =* 48.0
start_directory = Dynamic_Connection_to_DB *parent_version_no =* 125.12
step_text = UPDATE TPCH_AUX_TABLE SET updateset=updateset+1

step_label = Sequential Refresh Stream Execution *step_id =* 1208 *global_flag =* 0
sequence_no = 1 *step_level =* 1 *parent_step_id =* 1203 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 4.12
start_directory = *parent_version_no =* 49.3
step_text =

step_label = Parallel Stream Execution ***step_id =*** 1209 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 1 ***parent_step_id =*** 1203 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %MAX_STREAMS%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 7.0
start_directory = ***parent_version_no =*** 49.3
step_text =

step_label = Power - Execute Query 14 ***step_id =*** 1211 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\14.sql ***version_no =*** 21.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 02 ***step_id =*** 1212 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\2.sql ***version_no =*** 22.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 09 ***step_id =*** 1213 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\9.sql ***version_no =*** 20.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 20 ***step_id =*** 1214 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\20.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 06 ***step_id =*** 1215 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\6.sql ***version_no =*** 15.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 17 ***step_id =*** 1216 ***global_flag =*** 0
sequence_no = 6 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\17.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 18 ***step_id =*** 1217 ***global_flag =*** 0
sequence_no = 7 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\18.sql ***version_no =*** 18.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 08 ***step_id =*** 1218 ***global_flag =*** 0
sequence_no = 8 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\8.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 21 ***step_id =*** 1219 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\21.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 13 ***step_id =*** 1220 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\13.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 03 ***step_id =*** 1221 ***global_flag =*** 0
sequence_no = 11 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\3.sql ***version_no =*** 20.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 22 ***step_id =*** 1222 ***global_flag =*** 0
sequence_no = 12 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\22.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 16 ***step_id =*** 1223 ***global_flag =*** 0
sequence_no = 13 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\16.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 04 ***step_id =*** 1224 ***global_flag =*** 0
sequence_no = 14 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\4.sql ***version_no =*** 20.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 11 ***step_id =*** 1225 ***global_flag =*** 0
sequence_no = 15 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\11.sql ***version_no =*** 20.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 15 ***step_id =*** 1226 ***global_flag =*** 0
sequence_no = 16 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\15.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 01 ***step_id =*** 1227 ***global_flag =*** 0
sequence_no = 17 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\1.sql ***version_no =*** 18.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 10 ***step_id =*** 1228 ***global_flag =*** 0
sequence_no = 18 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\10.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 19 ***step_id =*** 1229 ***global_flag =*** 0
sequence_no = 19 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\19.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 05 ***step_id =*** 1230 ***global_flag =*** 0
sequence_no = 20 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\5.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 07 ***step_id =*** 1231 ***global_flag =*** 0
sequence_no = 21 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\7.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Power - Execute Query 12 ***step_id =*** 1232 ***global_flag =*** 0
sequence_no = 22 ***step_level =*** 2 ***parent_step_id =*** 1205 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\Power\12.sql ***version_no =*** 19.0
start_directory = Power_Static_DB_Connection ***parent_version_no =*** 26.0
step_text =

step_label = Throughput - Semaphore Loop for RF Delay ***step_id =*** 1234 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1208 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 6.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 4.12
step_text = %TOOLS_DIR%\Utility\semaphore -waitgroup S -count %MAX_STREAMS%

step_label = Throughput - Refresh Streams ***step_id =*** 1235 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1208 ***enabled_flag =*** -1
iterator_name = STREAM_NUM ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = ***parent_version_no =*** 4.12
step_text =

step_label = Stream 1 Manager ***step_id =*** 1236 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1209 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.3
start_directory = ***parent_version_no =*** 7.0
step_text =

step_label = Stream 2 Manager ***step_id =*** 1237 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1209 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.3
start_directory = ***parent_version_no =*** 7.0
step_text =

step_label = Stream 3 Manager ***step_id =*** 1238 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1209 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.8
start_directory = ***parent_version_no =*** 7.0
step_text =

step_label = Stream 4 Manager *step_id =* 1239 *global_flag =* 0
sequence_no = 4 *step_level =* 2 *parent_step_id =* 1209 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 1.3
start_directory = *parent_version_no =* 7.0
step_text =

step_label = Stream 5 Manager *step_id =* 1240 *global_flag =* 0
sequence_no = 5 *step_level =* 2 *parent_step_id =* 1209 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 1.3
start_directory = *parent_version_no =* 7.0
step_text =

step_label = Stream 6 Manager *step_id =* 1241 *global_flag =* 0
sequence_no = 6 *step_level =* 2 *parent_step_id =* 1209 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 3.3
start_directory = *parent_version_no =* 7.0
step_text =

step_label = Stream 7 Manager *step_id =* 1242 *global_flag =* 0
sequence_no = 7 *step_level =* 2 *parent_step_id =* 1209 *enabled_flag =* -1
iterator_name = *degree_parallelism* 1
execution_mechanism = 0 *continuation_criteria =* 0 *failure_details =*
step_file_name = *version_no =* 2.3
start_directory = *parent_version_no =* 7.0
step_text =

step_label = Stream 8 Manager ***step_id =*** 1243 ***global_flag =*** 0
sequence_no = 8 ***step_level =*** 2 ***parent_step_id =*** 1209 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 7.0
step_text =

step_label = Stream 9 Manager ***step_id =*** 1244 ***global_flag =*** 0
sequence_no = 9 ***step_level =*** 2 ***parent_step_id =*** 1209 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 7.0
step_text =

step_label = Stream 10 Manager ***step_id =*** 1245 ***global_flag =*** 0
sequence_no = 10 ***step_level =*** 2 ***parent_step_id =*** 1209 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 7.0
step_text =

step_label = Throughput - RF1 - Stream%STREAM_NUM% ***step_id =*** 1246 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1235 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %INSERT_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 2.0
start_directory = ***parent_version_no =*** 4.0
step_text =

step_label = Throughput - RF2 - Stream%STREAM_NUM% ***step_id =*** 1247 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 3 ***parent_step_id =*** 1235 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %DELETE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 4.0
step_text =

step_label = Throughput - Stream - Increment Update Set ***step_id =*** 1248 ***global_flag =*** 0
sequence_no = 5 ***step_level =*** 3 ***parent_step_id =*** 1235 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = Throughput_RF_Connection ***parent_version_no =*** 4.0
step_text = UPDATE TPCH_AUX_TABLE SET updateset=updateset+1

step_label = Throughput - Query Stream 1 ***step_id =*** 1249 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1236 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM1\Stream1Q%QUERY%.sql ***version_no =*** 1.0
start_directory = Throughput_Static_Stream_1_Connection ***parent_version_no =*** 1.3
step_text =

step_label = Throughput - Post to Semaphore (S1) ***step_id =*** 1250 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1236 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 1.3
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.1

step_label = Throughput - Query Stream 2 ***step_id =*** 1251 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1237 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM2\Stream2Q%QUERY%.sql ***version_no =*** 1.0
start_directory = Throughput_Static_Stream_2_Connection ***parent_version_no =*** 1.3
step_text =

step_label = Throughput - Post to Semaphore (S2) ***step_id =*** 1252 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1237 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 3.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 1.3
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.2

step_label = Throughput - Query Stream 3 ***step_id =*** 1253 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1238 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM3\Stream3Q%QUERY%.sql ***version_no =*** 4.0
start_directory = Throughput_Static_Stream_3_Connection ***parent_version_no =*** 0.8
step_text =

step_label = Throughput - Post to Semaphore (S3) ***step_id =*** 1254 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1238 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 2.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.8
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.3

step_label = Throughput - Query Stream 4 ***step_id =*** 1255 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1239 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM4\Stream4Q%QUERY%.sql ***version_no =*** 2.0
start_directory = Throughput_Static_Stream_4_Connection ***parent_version_no =*** 1.3
step_text =

step_label = Throughput - Post to Semaphore (S4) ***step_id =*** 1256 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1239 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 3.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 1.3
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.4

step_label = Throughput - Query Stream 5 ***step_id =*** 1257 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1240 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM5\Stream5Q%QUERY%.sql ***version_no =*** 1.0
start_directory = Throughput_Static_Stream_5_Connection ***parent_version_no =*** 1.3
step_text =

step_label = Throughput - Post to Semaphore (S5) ***step_id =*** 1258 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1240 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 3.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 1.3
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.5

step_label = Throughput - Query Stream 6 ***step_id =*** 1259 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1241 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM6\Stream6Q%QUERY%.sql ***version_no =*** 1.0
start_directory = Throughput_Static_Stream_6_Connection ***parent_version_no =*** 3.3
step_text =

step_label = Throughput - Post to Semaphore (S6) ***step_id =*** 1260 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1241 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 2.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 3.3
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.6

step_label = Throughput - Query Stream 7 ***step_id =*** 1261 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1242 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM7\Stream7Q%QUERY%.sql ***version_no =*** 1.0
start_directory = Throughput_Static_Stream_7_Connection ***parent_version_no =*** 2.3
step_text =

step_label = Throughput - Post to Semaphore (S7) ***step_id =*** 1262 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1242 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 2.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 2.3
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.7

step_label = Throughput - Query Stream 8 ***step_id =*** 1263 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1243 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM8\Stream8Q%QUERY%.sql ***version_no =*** 0.0
start_directory = Throughput_Static_Stream_8_Connection ***parent_version_no =*** 1.0
step_text =

step_label = Throughput - Post to Semaphore (S8) ***step_id =*** 1264 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1243 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 1.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.8

step_label = Throughput - Query Stream 9 ***step_id =*** 1265 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1244 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM9\Stream9Q%QUERY%.sql ***version_no =*** 0.0
start_directory = Throughput_Static_Stream_9_Connection ***parent_version_no =*** 1.0
step_text =

step_label = Throughput - Post to Semaphore (S9) ***step_id =*** 1266 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1244 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 1.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.9

step_label = Throughput - Query Stream 10 ***step_id =*** 1267 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1245 ***enabled_flag =*** -1
iterator_name = QUERY ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %QUERY_DIR%\STREAM10\Stream10Q%QUERY%.sql ***version_no =*** 0.0
start_directory = Throughput_Static_Stream_10_Connection ***parent_version_no =*** 1.0
step_text =

step_label = Throughput - Post to Semaphore (S10) ***step_id =*** 1268 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 3 ***parent_step_id =*** 1245 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 2 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 1.0
step_text = %TOOLS_DIR%\Utility\semaphore -signal S.10

step_label = Throughput - Execute Stream%STREAM_NUM% RF1 ***step_id =*** 1269 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1246 ***enabled_flag =*** -1
iterator_name = INSERT_SEGMENT ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 5.0
start_directory = Throughput_RF_Connection ***parent_version_no =*** 2.0
step_text =
```
--
-- Execute the Refresh RF1 Stored Procedure
--
EXEC  RF1 %INSERT_SEGMENT%, %BATCH_SIZE%, %INSERT_PARALLELISM%,
%INSERT_EXECUTIONS%
```

step_label = Throughput - Execute Stream%STREAM_NUM% RF2 ***step_id =*** 1270 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1247 ***enabled_flag =*** -1
iterator_name = DELETE_SEGMENT ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 4.0
start_directory = Throughput_RF_Connection ***parent_version_no =*** 1.0
step_text =
```
--
-- Execute the Refresh RF2 Stored Procedure
--
EXEC RF2 %DELETE_SEGMENT%, %BATCH_SIZE%, %DELETE_PARALLELISM%,
%DELETE_EXECUTIONS%
```

step_label = SqlServer Startup ***step_id =*** 1294 ***global_flag =*** -1
sequence_no = 1 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 0
execution_mechanism = 2 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text = start "SqlServer" sqlservr %TRACEFLAGS%
%TOOLS_DIR%\Utility\wait4sql 40000

step_label = SqlServer Shutdown ***step_id =*** 1295 ***global_flag =*** -1
sequence_no = 2 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 0
execution_mechanism = 2 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.0
step_text = osql -Usa -P -t10 -Q"shutdown"
%TOOLS_DIR%\Utility\wait4sql 10000

step_label = Wait For SQL Server ***step_id =*** 1296 ***global_flag =*** -1
sequence_no = 3 ***step_level =*** 0 ***parent_step_id =*** 0 ***enabled_flag =*** 0
iterator_name = ***degree_parallelism*** 0
execution_mechanism = 2 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = %TOOLS_DIR% ***parent_version_no =*** 0.0
step_text = %TOOLS_DIR%\Utility\wait4sql.exe 60000

step_label = Power - Execute RF1 ***step_id =*** 1405 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 1 ***parent_step_id =*** 1202 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 16.0
start_directory = ***parent_version_no =*** 125.12
step_text =

step_label = Parallel RF1 Execution ***step_id =*** 1406 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 1405 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %INSERT_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 3.0
start_directory = ***parent_version_no =*** 16.0
step_text =

step_label = Execute RF1 ***step_id =*** 1407 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1406 ***enabled_flag =*** -1
iterator_name = INSERT_SEGMENT ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 9.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 3.0
step_text =

```
--
-- Execute the Refresh RF1 Stored Procedure
--

EXEC  RF1 %INSERT_SEGMENT%, %BATCH_SIZE%, %INSERT_parallelism%,
%INSERT_EXECUTIONS%
```

step_label = Create RF1 Tables ***step_id =*** 1411 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1405 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF1_init.sql ***version_no =*** 9.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 16.0
step_text =

step_label = Parallel Load RF1 Input ***step_id =*** 1414 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1405 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FILES_PER_UPDATE_SET%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 16.0
step_text =

step_label = Throughput - Load RF1 Input ***step_id =*** 1415 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 5 ***parent_step_id =*** 1431 ***enabled_flag =*** -1
iterator_name = SEGMENT ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF1_load.sql ***version_no =*** 3.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Create RF1 Indices ***step_id =*** 1417 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1405 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF1_index.sql ***version_no =*** 3.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 16.0
step_text =

step_label = Power - Execute RF2 ***step_id =*** 1419 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 1 ***parent_step_id =*** 1202 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 16.0
start_directory = ***parent_version_no =*** 125.12
step_text =

step_label = Create RF2 Tables ***step_id =*** 1420 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 2 ***parent_step_id =*** 1419 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF2_init.sql ***version_no =*** 4.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 16.0
step_text =

step_label = Parallel Load RF2 Input ***step_id =*** 1421 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 2 ***parent_step_id =*** 1419 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FILES_PER_UPDATE_SET%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 1.0
start_directory = ***parent_version_no =*** 16.0
step_text =

step_label = Load RF2 Input ***step_id =*** 1422 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1421 ***enabled_flag =*** -1
iterator_name = SEGMENT ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF2_load.sql ***version_no =*** 2.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 1.0
step_text =

step_label = Create RF2 Indices ***step_id =*** 1424 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 2 ***parent_step_id =*** 1419 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF2_index.sql ***version_no =*** 4.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 16.0
step_text =

step_label = Parallel RF2 Execution ***step_id =*** 1425 ***global_flag =*** 0
sequence_no = 4 ***step_level =*** 2 ***parent_step_id =*** 1419 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %DELETE_PARALLELISM%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 3.0
start_directory = ***parent_version_no =*** 16.0
step_text =

step_label = Execute RF2 ***step_id =*** 1426 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1425 ***enabled_flag =*** -1
iterator_name = DELETE_SEGMENT ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = ***version_no =*** 8.0
start_directory = Power_Dynamic_DB_Connection ***parent_version_no =*** 3.0
step_text =

```
--
-- Execute the Refresh RF2 Stored Procedure
--

EXEC RF2 %DELETE_SEGMENT%, %BATCH_SIZE%, %DELETE_PARALLELISM%,
%DELETE_EXECUTIONS%
```

step_label = Throughput - Init RF1 Input ***step_id =*** 1428 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1235 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 2.0
start_directory = ***parent_version_no =*** 4.0
step_text =

step_label = Throughput - Init RF2 Input ***step_id =*** 1429 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 3 ***parent_step_id =*** 1235 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.1
start_directory = ***parent_version_no =*** 4.0
step_text =

step_label = Throughput - Create RF1 Tables ***step_id =*** 1430 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1428 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 1 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF1_init.sql ***version_no =*** 7.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 2.0
step_text =

step_label = Throughput - Parallel Load RF1 Input ***step_id =*** 1431 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 4 ***parent_step_id =*** 1428 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FILES_PER_UPDATE_SET%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 2.0
step_text =

step_label = Load RF1 Input ***step_id =*** 1432 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 3 ***parent_step_id =*** 1414 ***enabled_flag =*** -1
iterator_name = SEGMENT ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF1_load.sql ***version_no =*** 3.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 1.0
step_text =

step_label = Throughput - Create RF1 Indices ***step_id =*** 1434 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 4 ***parent_step_id =*** 1428 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF1_index.sql ***version_no =*** 1.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 2.0
step_text =

step_label = Throughput - Create RF2 Tables ***step_id =*** 1435 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 4 ***parent_step_id =*** 1429 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF2_init.sql ***version_no =*** 4.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 0.1
step_text =

step_label = Throughput - Parallel Load RF2 Input ***step_id =*** 1436 ***global_flag =*** 0
sequence_no = 2 ***step_level =*** 4 ***parent_step_id =*** 1429 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** %FILES_PER_UPDATE_SET%
execution_mechanism = 0 ***continuation_criteria =*** 0 ***failure_details =***
step_file_name = ***version_no =*** 0.0
start_directory = ***parent_version_no =*** 0.1
step_text =

step_label = Throughput - Load RF2 Input ***step_id =*** 1437 ***global_flag =*** 0
sequence_no = 1 ***step_level =*** 5 ***parent_step_id =*** 1436 ***enabled_flag =*** -1
iterator_name = SEGMENT ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF2_load.sql ***version_no =*** 2.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 0.0
step_text =

step_label = Throughput - Create RF2 Indices ***step_id =*** 1438 ***global_flag =*** 0
sequence_no = 3 ***step_level =*** 4 ***parent_step_id =*** 1429 ***enabled_flag =*** -1
iterator_name = ***degree_parallelism*** 1
execution_mechanism = 1 ***continuation_criteria =*** 2 ***failure_details =***
step_file_name = %SETUP_DIR%\RF_Procs\RF2_index.sql ***version_no =*** 3.0
start_directory = Dynamic_Connection_to_DB ***parent_version_no =*** 0.1
step_text =

Iterator_values

workspace_id	*step_id*	*version_no*	*type*	*iterator_value*	*sequence_no*
13	1432	3.0	2	%FILES_PER_UPDATE_SET%	0
13	1422	2.0	1	1	0
13	1422	2.0	2	%FILES_PER_UPDATE_SET%	0
13	1422	2.0	3	1	0
13	1270	4.0	3	1	0
13	1270	4.0	1	1	0
13	1270	4.0	2	%DELETE_EXECUTIONS%	0

13	1437	2.0	2	%FILES_PER_UPDATE_SET%		0
13	1437	2.0	3	1		0
13	1437	2.0	1	1		0
13	1415	3.0	2	%FILES_PER_UPDATE_SET%		0
13	1415	3.0	1	1		0
13	1267	0.0	3	1		0
13	1432	3.0	1	1		0
13	1426	8.0	1	1		0
13	1432	3.0	3	1		0
13	1407	9.0	1	1		0
13	1407	9.0	3	1		0
13	1407	9.0	2	%INSERT_EXECUTIONS%		0
13	1265	0.0	2	22		0
13	1265	0.0	3	1		0
13	1265	0.0	1	1		0
13	1263	0.0	3	1		0
13	1263	0.0	2	22		0
13	1263	0.0	1	1		0
13	1267	0.0	2	22		0
13	1267	0.0	1	1		0
13	1415	3.0	3	1		0
13	1257	1.0	2	22		0
13	1235	4.0	2	%MAX_STREAMS%		0
13	1235	4.0	1	1		0
13	1249	1.0	1	1		0
13	1249	1.0	2	22		0
13	1249	1.0	3	1		0
13	1251	1.0	3	1		0
13	1251	1.0	1	1		0
13	1251	1.0	2	22		0

13	1253	4.0	1	1		0
13	1253	4.0	2	22		0
13	1253	4.0	3	1		0
13	1255	2.0	2	22		0
13	1426	8.0	2	%DELETE_EXECUTIONS%		0
13	1255	2.0	1	1		0
13	1426	8.0	3	1		0
13	1257	1.0	1	1		0
13	1257	1.0	3	1		0
13	1259	1.0	1	1		0
13	1259	1.0	2	22		0
13	1259	1.0	3	1		0
13	1261	1.0	3	1		0
13	1261	1.0	2	22		0
13	1261	1.0	1	1		0
13	1269	5.0	3	1		0
13	1269	5.0	2	%INSERT_EXECUTIONS%		0
13	1269	5.0	1	1		0
13	1235	4.0	3	1		0
13	1255	2.0	3	1		0

Appendix G : Disk Configuration

HBA#	storage system#	SP#	DEU#	# of Disks	RAID Level	Capacity (GB)	Disk Partition1	
Partitions for DB Log								
0	0	0	0	15	6	393	#1(10GB) z: for junction point	#2(383GB) log1 for tpch3t
			1	15	6	393		#2(383GB) log2 for tpch3t
			2	15	6	393		#2(383GB) log3 for tpch3t
			3	15	6	393	#1(393GB) templog	

HBA	storage system#	SP#	DEU#	# of Disks	RAID Level	Capacity (GB)	Partition			
1-1	1	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
1-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
2-1	2	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
2-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
3-1	3	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
3-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
4-1	4	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
4-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
5-1	5	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
5-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
6-1	6	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
6-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
7-1	7	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
7-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
8-1	8	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
8-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
9-1	9	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
9-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
10-1	10	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
10-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
11-1	11	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
11-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
12-1	12	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
12-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
13-1	13	0	0	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
13-2		1	1	15	0	499	#1(240GB):General	#2(120GB):Lineltem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			

14-1	14	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
14-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
15-1	15	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
15-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
16-1	16	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
16-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & flat files			
17-1	17	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
17-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup & update files			
18-1	18	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
18-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup			
19-1	19	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
19-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup			
20-1	20	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
20-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup			
21-1	21	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
21-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup			
22-1	22	0	0	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
22-2		1	1	15	0	499	#1(240GB):General	#2(120GB):LineItem	#3(100GB):temp	
			2	15	5	466	#1(466GB): backup			

Appendix H : Price Quotation

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

Tel 425 882 8080
Fax 425 936 7329
http://www.microsoft.com/

Microsoft

October 20, 2005

NEC Corporation
Keiichi Yamada
1-10 Nisshin-cho, Fuchu-shi
Tokyo, 1830035

Yamada-san:

Here is the information you requested regarding pricing for several Microsoft products to be used in conjunction with your TPC-H benchmark testing.

All pricing shown is in US Dollars ($).

Part Number	Description	Unit Price	Quantity	Price
810-04793	**SQL Server 2005 Enterprise Itanium Edition** *Server License with 25 CALs* *Discount Schedule: Open Program - No Level* *Unit Price reflects a 39% discount from the retail unit price of $13,969.*	$8,487	1	$8,487
359-01911	**SQL Server 2005 Client License** *Discount Schedule: Open Program - No Level* *Unit Price reflects a 4% discount from the retail unit price of $163.*	$156	65	$10,140
P73-00295	**Windows Server 2003 Standard Edition** *Server License Only - No CALs* *Discount Schedule: Open Program - No Level* *Unit Price reflects a 28% discount from the retail unit price of $999.*	$719	1	$719
659-00390	**Visual Studio .Net 2003 Professional** *No Discount Applied*	$799	1	$799
N/A	**Microsoft Problem Resolution Services** *Professional Support* *(1 Incident)*	$245	1	$245

Some products may not be currently orderable but will be available through Microsoft's normal distribution channels by November 7, 2005.

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at $245 per call.

This quote is valid for the next 90 days.

If we can be of any further assistance, please contact Jamie Reding at (425) 703-0510 or jamiere@microsoft.com.

Reference ID: PHkeya0520103843.
Please include this Reference ID in any correspondence regarding this price quote.

compuview Microsystems, Inc.

VIEW CART | SUPPORT encoreSHOP.com A Division of Compuview

HOME | COMPUTERS | SOFTWARE | ELECTRONICS | PHOTOGRAPHY | OFFICE | GAMES

Tuesday , November 01, 2005.

Power Search | Order Status

Shopping Cart

	Quantity	Item Name	Price	Extension
Remove	25	NET/QLA2342-CK-BK Qlogic 64-bit 133MHz PCI-X to 2Gb Fibre Channel adapter QLA2342-CK	$1,395.00	$34,875.00
Remove	2	VL5103-BX CTX VL5103 VL510 (15", 1024x768, .28mm)	$94.12	$188.24
Recalculate			SubTotal	$35,063.24
Shipping Method:	Other	Calculate Shipping		$0.00

(Order multiple items may have shipping discount please call in for information. Select "Other" in the shipping method to use your carrier account, $5 handling fee may be applied.)

Coupon Code: Submit

Please note that 8.25% sales Tax will be added for Texas residents. **Download Tax Exemption Form (PDF)**
If the package is going to the Islands (PR Puerto Rico, Hawaii, Alaska, Virgin Islands) of United States, Please choose 2 days or 1 day Air shipping. Otherwise, your order may be delayed in processing. Shipping cost will be slightly higher than the price listed. All orders placed after 2:00PM CST are not guaranteed being shipped the same day. Please call in to confirm.

Continue Shopping | Proceed to Checkout

Terms of Policies | Privacy | Job Openings | News | Contact | About us