

---

**TPC Benchmark™ H**  
**Full Disclosure Report**

**HITACHI BladeSymphony BS1000**  
**(8 SMP)**

**with Microsoft SQL Server 2005**  
**Enterprise Edition for Itanium SP1**  
**and**  
**Microsoft® Windows® Server 2003, Enterprise Edition**  
**for Itanium SP1**

---

**Second Edition**

**Submitted for Review**  
**May 17, 2006**

**Revision date**  
**July 21, 2006**

---

Hitachi, Ltd. (HITACHI), 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. HITACHI 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 HITACHI HARDWARE PRODUCTS AND THE LICENSING OF HITACHI SOFTWARE CONSIST SOLELY OF THOSE SET FORTH IN THE WRITTEN CONTRACTS BETWEEN HITACHI 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 HITACHI FOR ANY PURPOSE, OR GIVE RISE TO ANY LIABILITY OF HITACHI WHATSOEVER.

HITACHI assumes no responsibility for any errors that may appear in this document. HITACHI 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 HITACHI to determine whether any such changes have been made.

Copyright 2006 Hitachi, Ltd.  
All rights reserved.

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

BladeSymphony is registered trademark of **Hitachi, Ltd.**

Microsoft, Windows Server 2003 and SQL Server 2005 are registered trademarks of **Microsoft Corporation.**

TPC Benchmark, TPC-H and QphH are trademarks of the **Transaction Processing Performance Council.**

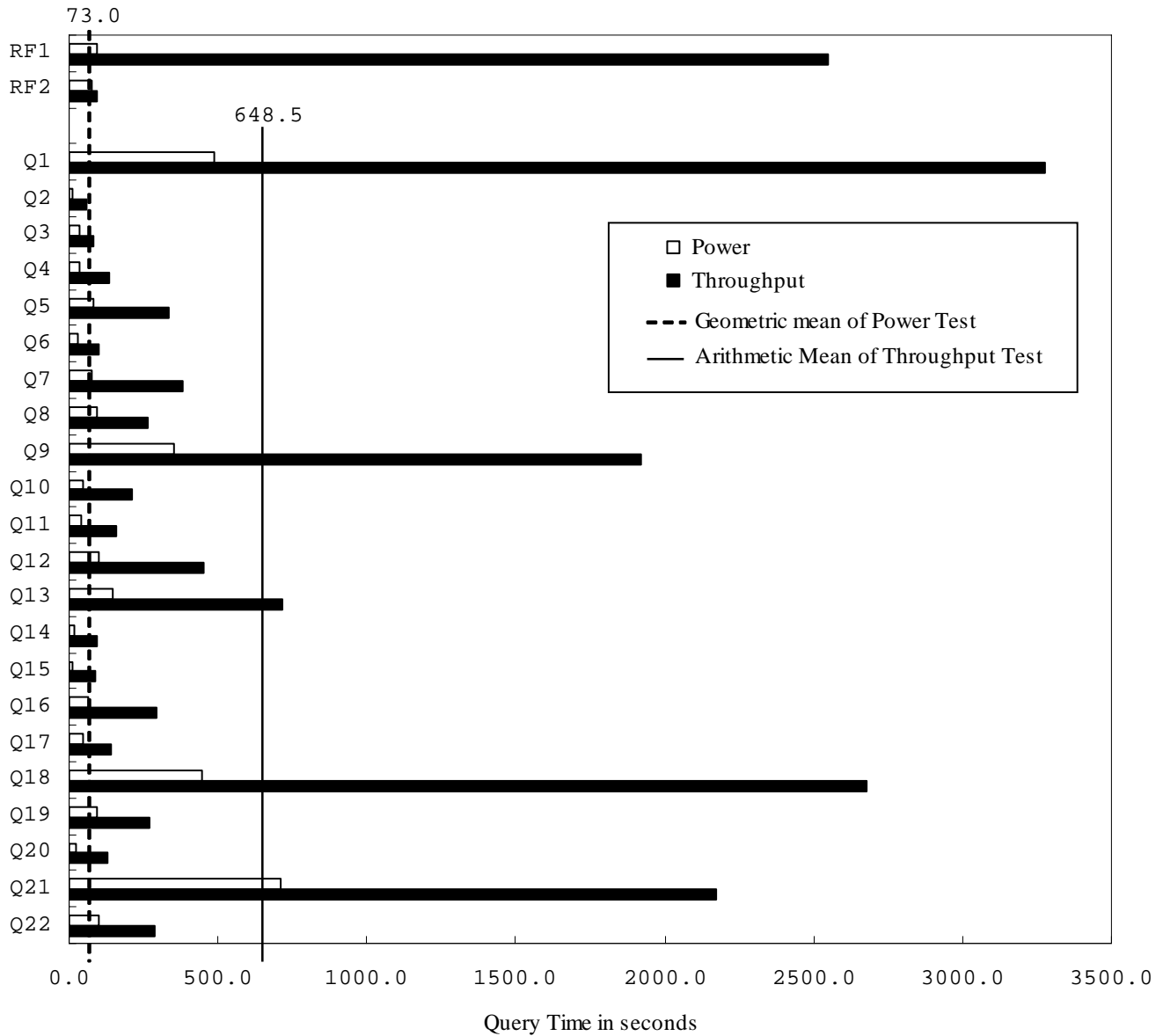
Intel, Itanium and Xeon are registered trademarks and trademark of **Intel Corporation.**

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

---

<b>HITACHI</b> Inspire the Next	<b>Hitachi BladeSymphony BS1000</b>	TPC-H Rev.2.3.0
		Report Date: May. 17, 2006 Revision Date: Jul. 21, 2006
Total System Cost	Composite Query per Hour Metric	Price/Performance
<b>\$810,428 (USD)</b>	<b>11,527.4 QphH@300GB</b>	<b>\$70.30 / QphH@300GB</b>

Database Size	Database Manager	Operating System	Other Software	Availability Date
<b>300GB</b>	<b>Microsoft® SQL Server™ 2005 Enterprise Edition for Itanium-Based Systems SP1</b>	<b>Microsoft® Windows Server™ 2003, Enterprise Edition for Itanium-based Systems SP1</b>	<b>Microsoft® Visual C++® Standard Edition, Microsoft® Visual Basic® 2003 .Net™ Professional</b>	<b>May 9, 2006</b>



Database Load Time = 07:25:45	Load included Backup: Y	Total Data Storage/Database Size = 34.80
RAID(Base Table Only): N	RAID(Base tables and auxiliary data structures): N	RAID(All): N

<b>System Configuration</b>	
<b>Processors/Cores/Threads</b>	8/8/8 1.66GHz Intel® Itanium®2 with 9MB L3 cache
<b>Memory</b>	128GB main memory
<b>Disk Controllers</b>	16x 4Gb FC HBA
<b>Disk Drives</b>	290x 36GB 15krpm FC
<b>Total Disk Storage</b>	10,440GB

Note: Database Size includes only raw data (e.g., no temp, index, redundant storage space, etc).

Description	Part Number	Third Party		Unit Price	Qty	Extended Price	3-Yr.Maint. Price	
		Brand	Pricing					
<b>Server Hardware</b>								
Chassis BS1000B (A2)	GVX-SREA2B51N	HAL		1	6,545	1	6,545	
IPF 2-way Server Module Itanium2 1.66GHz/9MB x2, FSB667	GVAX75A2-7NN0N5X	HAL		1	16,147	4	64,588	
8 GB Memory (DDR2)	GVX-MJ908GAEX	HAL		1	3,790	16	60,640	
IPF SMP Upgrade License (2way to 4way) (Itanium2 1.66GHz/9MB)		HAL		1	1,808	1	1,808	
IPF SMP Upgrade License (4way to 8way) (Itanium2 1.66GHz/9MB)		HAL		1	155,676	1	155,676	
I/O Module TYPE-1	GVX-BE9SLT1BX	HAL		1	1,558	2	3,116	
4-Gbps FC Adapter (2 Ch)	GVX-CC64G2BX	HAL		1	590	16	9,440	
Power Supply Module	GVX-BP9200VBX	HAL		1	265	2	530	
Dummy Server Module	GVX-BE9BLDMYBX	HAL		1	39	3	117	
Dummy SVP Module	GVX-BE9SVPDMYBX	HAL		1	82	1	82	
Xeon Server Module Xeon 3.D GHz/1MB x1	GVAX51A2-1NN0N4X	HAL		1	2,407	1	2,407	
1 GB Memory (DDR1)	GVX-MJ701GTEX	HAL		1	277	1	277	
HDD 160 GB	GVX-UH7160NEX	HAL		1	129	2	258	
16 U Rack Cabinet Set	GVX-SRK7165BK	HAL		1	1,046	1	1,046	
16 U Front Door BY	GVX-RD9166BK	HAL		1	384	1	384	
16 U Rack Side Panel	GVX-RS7165BK	HAL		1	147	1	147	
Blank Panel	GVX-RP7011BK	HAL		1	6	2	12	
Outlet Box 3N	GVX-AG9203A	HAL		1	675	1	675	
KVM Cable	GVX-LY7201	HAL		1	70	1	70	
17" SXGA Console Drawer FD-1000 series	NC14010-B452/US	HAL		1	2,493	1	2,493	
KVM Cascade Cable (USB)	NC14000-B102	HAL		1	35	1	35	
Guide Rail	NC14003-T591	HAL		1	52	1	52	
Floppy Drive External USB Floppy Drive	DC361B	HAL		1	73	1	73	
24X External DVD/CDRW USB2.0	EZ-WTR-DVD-C24	HAL		1	158	1	158	
3 year maintenance service(Express)		HAL		1	210,733	1	210,733	
Subtotal						310,629	210,733	
<b>Server Storage</b>								
9500V Storage Expansion Unit (RKA)	DF600-RKA	HDS		2	6,300	22	138,600	
9585V Base Unit(RKHE) w/2 controllers 4 FC ports	DF600-RKHE	HDS		2	38,000	2	76,000	
9500V 72GB FC 15K RPM HDD	DF-F600-AEH72	HDS		2	1,410	328	462,480	
9580V 2GB Cache Memory Module 2x1024MB	DF-F600-C1GX2H	HDS		2	4,830	8	38,640	
9580V Fibre Channel Intfc 2Gbps w/2 FC ports	DF-F600-DH2G2	HDS		2	3,800	4	15,200	
LC/LC Duplex FibreOptic Plenum Cable 5M	JZ-050LL005PC.P	HDS		2	185	16	2,960	
9570/85V Resource Manager License	044-210195-03.P	HDS		2	3,500	2	7,000	
9500V 125V AC 15A Power Cord	DF-F600-J1H	HDS		2	39	48	1,872	
Filler Panel 1.0U perf. for HDS Rack	122296-1-P.P	HDS		2	37	4	148	
19inch Rack for HDS-US	7846400.P	HDS		2	4,924	2	9,848	
Universal Rails (Kit)	7846406.P	HDS		2	84	24	2,016	
Installation		HDS		2	2,350	1	2,350	
3 year maintenance service(Standard support )		HDS		2	53,265	1	53,265	
Subtotal						757,114	53,265	
<b>Server Software</b>								
SQL Server 2005 Enterprise Itanium Edition SP1	810-04793	Microsoft		3	8,487	1	8,487	
SQL Server 2005 Client License	359-01911	Microsoft		3	156	35	5,460	
Windows Server 2003 Enterprise Edition for 64bit Itanium	P72-00284	Microsoft		3	2,334	1	2,334	
Windows Server 2003 Standard Edition	P73-00295	Microsoft		3	719	1	719	
Visual C++ Standard Edition	254-00170	Microsoft		3	109	1	109	included
Visual Basic 2003 .Net Professional	046-00856	Microsoft		3	109	1	109	included
Microsoft Problem Resolution Services		Microsoft		3	245	1	245	
Subtotal						17,218	245	
Total System Discounts						419,601	119,175	
Total						665,360	145,068	

Notes:

Pricing: 1-Hitachi America, Ltd (HAL), 2-Hitachi DATA Systems Corporation (HDS), 3-Microsoft  
Results and methodology audited by Francois Raab of InfoSizing, Inc. (www.sizing.com)

Three-Year Cost of Ownership: 810,428  
QphH @ 300GB: 11,527.4  
\$/ QphH @ 300GB: 70.30

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	300 GB
Total Data Storage/Database Size	34.80
Start of Database Load	4/19/06 16:35:33
End of Database Load	4/20/06 00:01:18
Database Load Time	07:25:45
Query Streams for Throughput Test	6
TPC-H Power	14,799.0
TPC-H Throughput	8,979.0
Composite Query per Hour Rating(QpH@300GB)	11,527.4
Total System Price Over 3 Years	\$810,428
TPC-H Price Performance Metric	\$70.30

### Measurement Intervals

Measurement Interval in Throughput Test(Ts)	15,877.0 seconds
---	------------------

### Duration of Stream Execution

	Seed	Query Start Date/Time		RF1 Start Date/Time		RF2 Start Date/Time		Query Duration
		Query End Date/Time	RF1 End Date/Time	RF2 End Date/Time	RF Duration			
<b>Stream0</b>	420000118	04/20/2006 10:00:22	04/20/2006 09:58:48	04/20/2006 10:51:22	04/20/2006 10:51:22	04/20/2006 10:51:22	04/20/2006 10:51:22	0:53:52
		04/20/2006 10:51:21	04/20/2006 10:00:22	04/20/2006 10:52:40	04/20/2006 10:52:40	04/20/2006 10:52:40	04/20/2006 10:52:40	
<b>Stream1</b>	420000119	04/20/2006 10:52:41	04/20/2006 10:52:41	04/20/2006 14:57:48	04/20/2006 14:57:48	04/20/2006 14:57:48	04/20/2006 14:57:48	3:55:55
		04/20/2006 14:48:36	04/20/2006 14:57:47	04/20/2006 14:59:11	04/20/2006 14:59:11	04/20/2006 14:59:11	04/20/2006 14:59:11	
<b>Stream2</b>	420000120	04/20/2006 10:52:41	04/20/2006 14:59:13	04/20/2006 15:01:00	04/20/2006 15:01:00	04/20/2006 15:01:00	04/20/2006 15:01:00	4:00:46
		04/20/2006 14:53:27	04/20/2006 15:01:00	04/20/2006 15:02:22	04/20/2006 15:02:22	04/20/2006 15:02:22	04/20/2006 15:02:22	
<b>Stream3</b>	420000121	04/20/2006 10:52:41	04/20/2006 15:02:24	04/20/2006 15:04:12	04/20/2006 15:04:12	04/20/2006 15:04:12	04/20/2006 15:04:12	3:54:38
		04/20/2006 14:47:19	04/20/2006 15:04:11	04/20/2006 15:05:44	04/20/2006 15:05:44	04/20/2006 15:05:44	04/20/2006 15:05:44	
<b>Stream4</b>	420000122	04/20/2006 10:52:41	04/20/2006 15:05:47	04/20/2006 15:07:39	04/20/2006 15:07:39	04/20/2006 15:07:39	04/20/2006 15:07:39	4:00:28
		04/20/2006 14:53:09	04/20/2006 15:07:38	04/20/2006 15:09:19	04/20/2006 15:09:19	04/20/2006 15:09:19	04/20/2006 15:09:19	
<b>Stream5</b>	420000123	04/20/2006 10:52:42	04/20/2006 15:09:22	04/20/2006 15:11:16	04/20/2006 15:11:16	04/20/2006 15:11:16	04/20/2006 15:11:16	3:51:26
		04/20/2006 14:44:08	04/20/2006 15:11:14	04/20/2006 15:13:04	04/20/2006 15:13:04	04/20/2006 15:13:04	04/20/2006 15:13:04	
<b>Stream6</b>	420000124	04/20/2006 10:52:42	04/20/2006 15:13:08	04/20/2006 15:15:24	04/20/2006 15:15:24	04/20/2006 15:15:24	04/20/2006 15:15:24	4:03:28
		04/20/2006 14:56:10	04/20/2006 15:15:22	04/20/2006 15:17:18	04/20/2006 15:17:18	04/20/2006 15:17:18	04/20/2006 15:17:18	

**TPC-H Timing Intervals(in seconds):**

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream0	485.2	10.3	37.2	37.6	80.6	32.0	79.0	95.2
Stream1	3457.2	74.2	87.9	101.8	507.3	85.2	397.7	247.5
Stream2	3184.5	61.0	98.0	128.8	286.5	115.2	633.8	271.6
Stream3	3178.5	59.8	70.0	173.3	365.4	114.2	327.1	290.4
Stream4	3286.1	67.9	50.4	140.7	278.3	90.3	294.0	234.5
Stream5	3354.3	54.1	114.7	142.1	269.6	96.5	308.3	229.5
Stream6	3197.8	44.7	85.7	138.4	314.7	114.8	318.8	322.3
Min Qi	3178.5	44.7	50.4	101.8	269.6	85.2	294.0	229.5
Max Qi	3457.2	74.2	114.7	173.3	507.3	115.2	633.8	322.3
Avg Qi	3276.4	60.3	84.5	137.5	337.0	102.7	380.0	266.0
Query	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
Stream0	354.1	44.4	42.9	102.2	145.5	17.9	14.1	64.4
Stream1	1518.2	194.5	222.4	380.0	651.0	94.3	118.6	279.0
Stream2	1679.7	249.8	172.1	639.7	693.1	99.5	80.4	357.1
Stream3	2105.4	157.2	152.8	399.3	714.2	98.9	65.4	303.3
Stream4	1797.1	213.1	129.7	467.4	752.9	90.6	75.7	269.8
Stream5	1930.3	204.5	159.3	394.5	654.3	99.1	83.1	281.4
Stream6	2496.0	251.1	127.8	446.1	833.9	84.9	89.7	269.8
Min Qi	1518.2	157.2	127.8	380.0	651.0	84.9	65.4	269.8
Max Qi	2496.0	251.1	222.4	639.7	833.9	99.5	118.6	357.1
Avg Qi	1921.1	211.7	160.7	454.5	716.6	94.6	85.5	293.4
Query	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream0	44.2	447.0	91.7	25.2	709.6	99.2	94.1	77.9
Stream1	144.8	2448.6	241.3	135.9	2482.9	284.6	14706.3	83.2
Stream2	174.8	2614.3	303.6	140.8	2169.5	292.2	107.0	81.3
Stream3	156.9	2690.5	248.0	90.1	1996.4	320.7	106.9	92.8
Stream4	126.9	3009.5	296.7	162.4	2339.6	253.8	110.5	100.9
Stream5	101.9	2194.4	268.1	107.0	2481.9	356.9	112.1	108.6
Stream6	137.5	3127.4	280.0	151.8	1558.9	216.5	134.3	114.3
Min Qi	101.9	2194.4	241.3	90.1	1558.9	216.5	106.9	81.3
Max Qi	174.8	3127.4	303.6	162.4	2482.9	356.9	14706.3	114.3
Avg Qi	140.5	2680.8	273.0	131.3	2171.5	287.5	2546.2	96.9

Benchmark Sponsor: Hitachi, Ltd.  
 Tadashi Nota  
 1 Horiyamashita,  
 Hadano-shi,  
 Kanagawa 259-1392, Japan

May 11, 2006

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

Platform: **Hitachi BladeSymphony BS1000**  
 Database Manager: **Microsoft SQL Server 2005 Enterprise Edition for Itanium-SP1**  
 Operating System: **Microsoft Windows Server 2003 Enterprise Edition for Itanium SP1**

The results were:

CPU (Speed)	Memory	Disks	QphH@300GB
<b>Hitachi BladeSymphony BS1000</b>			
8 x Intel Itanium2 (1.66 GHz)	9 MB L3 Cache 128 GB Main	290 x 36 GB 15 KRPM FC.	<b>11,527.4</b>

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 300GB 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 6 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:

The measured system was configured with 36 GB disks drives that will soon reach end of life. They were substituted by newer 72 GB disks, in the priced configuration. Based on the specifications of these disks it is my opinion that this substitution has no significant effect on performance.

Respectfully Yours,

A handwritten signature in black ink, appearing to read "François Raab", with a long horizontal flourish extending to the right.

François Raab  
President



## Table of Contents

PREFACE .....	1
Document Overview .....	1
TPC Benchmark™ H Overview .....	1
The TPC-H operations are modeled as follows: .....	2
General Implementation Guidelines .....	3
Related Product Information .....	3
1. GENERAL ITEMS .....	4
1.1. Benchmark Sponsor .....	4
1.2. Parameter Settings .....	4
1.3. Configuration Diagrams .....	4
2. CLAUSE 1: Logical Database Design .....	7
2.1. Table Definitions .....	7
2.2. Database Organization .....	7
2.3. Horizontal Partitioning .....	7
2.4. Replication .....	7
3. CLAUSE 2: QUERIES AND UPDATE FUNCTIONS .....	8
3.1. Query Language .....	8
3.2. Random Number Generation .....	8
3.3. Substitution Parameters .....	8
3.4. Query Text And Output Data from Qualification Database .....	8
3.5. Query Substitution Parameters and Seeds .....	8
3.6. Query Isolation Level .....	9
3.7. Source Code of Refresh Functions .....	9
4. CLAUSE 3: DATABASE SYSTEM PROPERTIES .....	10
4.1. Atomicity Requirements .....	10
4.1.1. Atomicity of Completed Transactions .....	10
4.1.2. Atomicity of Aborted Transactions .....	10
4.2. Consistency Requirements .....	10
4.2.1. Consistency Tests .....	10
4.3. Isolation Requirements .....	11
4.3.1. Isolation Test 1: Read-Write Conflict with Commit .....	11
4.3.2. Isolation Test 2: Read-Write Conflict with Rollback .....	11
4.3.3. Isolation Test 3: Write-Write Conflict with Commit .....	11
4.3.4. Isolation Test 4: Write-Write Conflict with Rollback .....	12
4.3.5. Isolation Test 5: Concurrent Progress of Read and Write on Different Tables .....	12
4.3.6. Isolation Test 6: Read-Only Query Conflict with Update Transaction .....	12
4.4. Durability Requirements .....	12
4.4.1. Loss of Data, Log .....	13
4.4.2. Loss of System .....	13
4.4.3. Loss of Memory .....	13
4.4.4. Loss of Controller .....	13
5. CLAUSE 4 : Scaling and Database Population .....	15
5.1. Cardinality of Tables .....	15
5.2. Distribution of Tables and Logs Across Media .....	15
5.3. Partitions/Replications Mapping .....	15
5.4. Use of RAID .....	15
5.5. DBGEN Modifications .....	15
5.6. Database Load Time .....	16
5.7. Database Storage Ratio .....	16

---

5.8.	Database Loading.....	16
5.9.	Qualification Database Configuration.....	16
6.	CLAUSE 5: Performance Metrics and Execution Rules .....	17
6.1.	Power Test Implementation .....	17
6.2.	Timing Intervals and Reporting .....	17
6.3.	Number of Streams in the Throughput Test.....	17
6.4.	Start and End Date/Time for Each Query Stream .....	17
6.5.	Total Elapsed Time for the Measurement Interval.....	17
6.6.	Refresh Function Start Date/Time and Finish Date/Time.....	17
6.7.	Timing Intervals for Each Query and Each Refresh Function for Each Stream .....	18
6.8.	Performance Metrics .....	18
6.9.	The Performance Metric and Numerical Quantities from Both Runs.....	18
6.10.	System Activity Between Tests .....	20
7.	CLAUSE 6 : SUT and Driver Implementation Related Items .....	21
7.1.	Driver .....	21
7.2.	Implementation-Specific Layer (ISL).....	21
8.	CLAUSE 7: PRICING RELATED ITEMS .....	23
8.1.	HARDWARE AND SOFTWARE USED .....	23
8.2.	Three-Year Cost of System Configuration.....	24
8.3.	Availability Dates.....	24
8.4.	Country-Specific Pricing .....	24
9.	CLAUSE 8: AUDIT RELATED ITEMS .....	25

APPENDIX A : System and Database Tunable Parameters

APPENDIX B : DATABASE, TABLE AND INDEXES CREATION

APPENDIX C : QUERY TEXT & OUTPUT

APPENDIX D : SEED & QUERY SUBSTITUTION

APPENDIX E : REFRESH FUNCTION SOURCE CODE

APPENDIX F : IMPLEMENTATION SPECIFIC LAYER AND SOURCE CODE

F.1 Source Code

F.1.1 print.c for DBGGEN

F.1.2 StepMaster

F.2 Configuration

APPENDIX G : DISK CONFIGURATION

APPENDIX H : PRICE QUOTATION

## PREFACE

### Document Overview

This report documents the methodology and results of the TPC Benchmark<sup>TM</sup> H (TPC-H) test conducted on the Hitachi BladeSymphony BS1000 using Microsoft SQL Server 2005 Enterprise Edition for Itanium-SP1, in conformance with the requirements of the TPC Benchmark<sup>TM</sup> H Standard Specification Revision 2.3.0. The tests documented in this report were sponsored by Hitachi, Ltd. The operating system used for the benchmark was Microsoft Windows Server 2003 Enterprise Edition for Itanium SP1. The Transaction Processing Performance Council (TPC) developed the TPC-H Benchmark. The TPC Benchmark<sup>TM</sup> H Standard represents an effort by Hitachi, Ltd. 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<sup>TM</sup> 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<sup>TM</sup> H Overview*

The TPC Benchmark<sup>TM</sup> 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., 100 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 TPC-H 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](mailto:info@tpc.org)

## 1. GENERAL ITEMS

### 1.1. Benchmark Sponsor

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

Hitachi, Ltd. is the sponsor of this TPC-H benchmark. The benchmark implementation was developed by Microsoft and Hitachi. The benchmark was conducted at Hitachi in Hadano-shi, Kanagawa.

### 1.2. Parameter Settings

*Settings must be provided for all customer-tunable parameters and options that 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

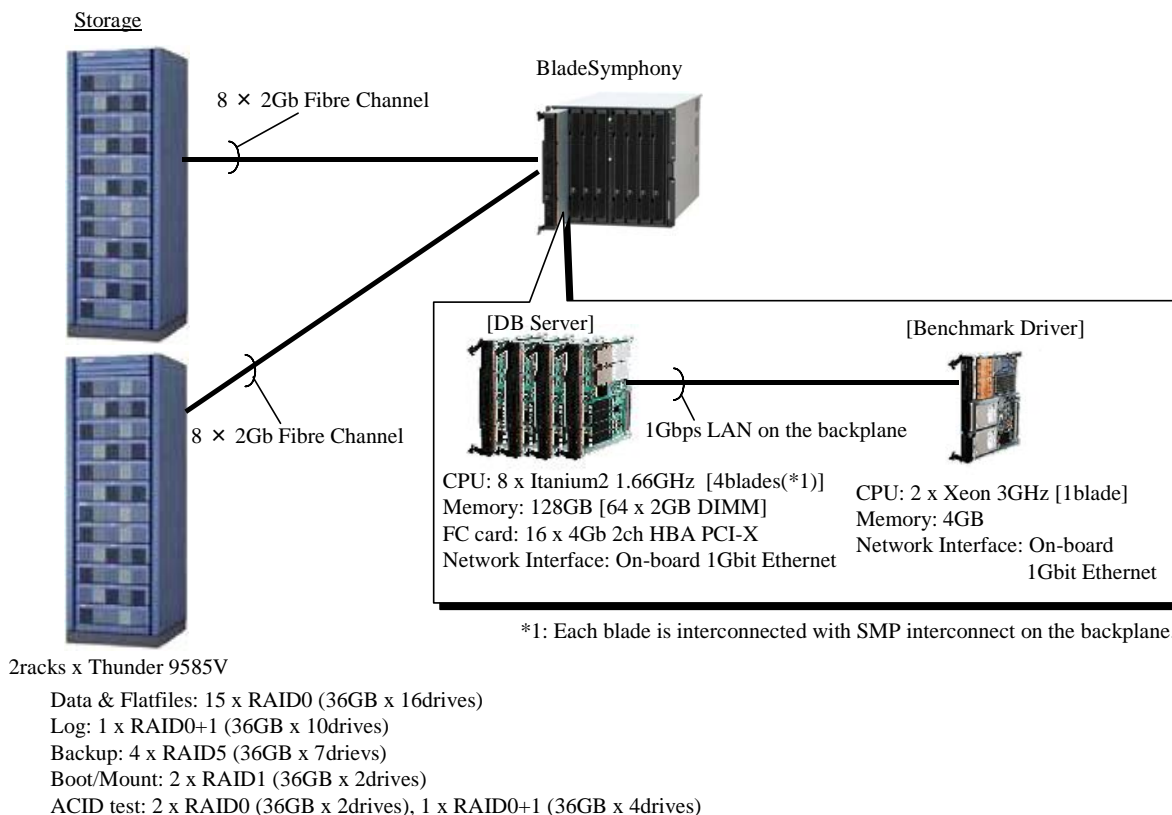
*Diagrams of both measured and priced configurations must be provided, accompanied by a description of the differences. This includes, but is not limited to:*

- Total number of nodes used, total number and type of processors used/total number of cores used/total number of threads used (including sizes of L2 and L3 caches);
- 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, middleware components, software drivers, etc.).

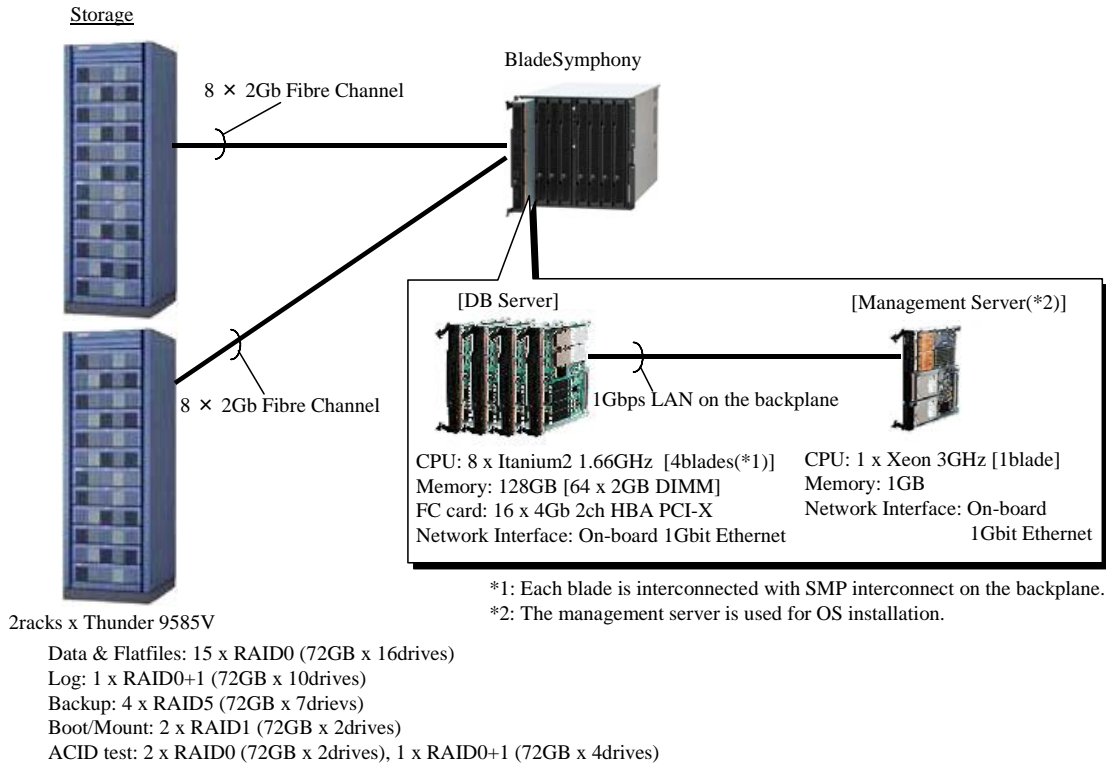
A priced configuration is different from the tested configuration regarding disk drives. For the priced configuration, 72 GB FC 15K RPM disks are used as a substituted component, because 36 GB FC 15K RPM disks will soon reach end of life. See section 8.1 for more details.

The configuration diagrams for the tested and priced system are provided on the following page.

**Figure 1-1: BladeSymphony BS1000, Tested Configurations**



**Figure 1-2: BladeSymphony BS1000, Priced Configurations**





## 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 tables and rows in the test and qualification database (see Clause) must be disclosed.*

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

### 2.4. Replication

*Any replication of physical objects must be disclosed and must conform to the requirements of Clause 1.5.6.*

Replication was not 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 2.3.0 and QGEN version 2.3.0 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 2.3.0 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).*

Appendix E contains the source code for the refresh functions.

## 4. CLAUSE 3: DATABASE SYSTEM PROPERTIES

*The results of the ACID tests must be disclosed along with a description of how the ACID requirements were met. This includes disclosure of the code written to implement the ACID Transaction and Query.*

### 4.1. Atomicity Requirements

*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. Atomicity of Completed Transactions

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

1. The total price from the ORDERS 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 ORDERS 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 changed.

#### 4.1.2. Atomicity of Aborted Transactions

*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 ORDERS, LINEITEM, and HISTORY tables.*

1. The total price from the ORDERS 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 ORDERS 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 Requirements

*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 Tests

*Verify that the ORDERS 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 ORDERS and LINEITEM tables.*

---

*The consistency of the ORDERS 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 ORDERS and LINEITEM tables.
2. More than 100 ACID transactions were submitted from each of 7 execution streams.
3. ACID queries were re-executed to verify the consistent state of the ORDERS and LINEITEM tables after the ACID transaction streams.
4. The consistency of the ORDERS and LINEITEM tables was re-verified.

#### 4.3. Isolation Requirements

*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. Isolation Test 1: Read-Write Conflict with Commit

*Demonstrate isolation for the read-write conflict of a read-write transaction and a read-only transaction when the 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. Isolation Test 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 DELTA. 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. Isolation Test 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 stopped 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
 
$$T2.L\_EXTENDEDPRICE = T1.L\_EXTENDEDPRICE + (\Delta T1 * (T1.L\_EXTENDEDPRICE / T1.L\_QUANTITY)).$$

#### 4.3.4. Isolation Test 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 stopped 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. Isolation Test 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. The ACID transaction, T1, was suspended prior to COMMIT.
2. Another ACID transaction, T2, was started using random values for PS\_PARTKEY and PS\_SUPPKEY.
3. T2 completed and T1 completed.
4. It was verified that the appropriate rows in the ORDERS, LINEITEM and HISTORY tables have been changed.

#### 4.3.6. Isolation Test 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 ORDERS, LINEITEM and HISTORY tables have been changed.

#### 4.4. Durability Requirements

*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 below.*

- Failure of a Durable Medium
- Failure of a System
- Failure of a Memory
- Failure of a Controller

#### 4.4.1. Loss of Data, 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 data.*

The database logs were stored on a RAID-10. The tables for the database were stored on RAID-0 stripes.

1. 7 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-10 set of log part disks was removed. The running continued without any interruptions.
3. Next, one of the RAID-0 set of data part disks was removed and a checkpoint submitted. Errors followed and the test was halted.
4. The transaction log was backup and the system rebooted replacing the failed drive.
5. The database was restored from data and log backups.
6. The counts in the success files and the HISTORY table counts were compared and the counts matched.

#### 4.4.2. Loss of System

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

The system crash and memory failure tests were combined.

1. 7 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 counts were compared and the counts matched.

#### 4.4.3. Loss of Memory

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

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

#### 4.4.4. Loss of Controller

*Guarantee the database and committed updates are preserved across a failure of all or part of*

---

*Controller.*

1. 7 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 checkpoint submitted, and then the controller was reset.
3. When the controller was recovered, and the database was restarted.
4. The database ran through its recovery mode.
5. The counts in the success files and the HISTORY table counts were compared and the counts matched.



## 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 Name	Row Count
Lineitem	1,799,989,091
Orders	450,000,000
Partsupp	240,000,000
Part	60,000,000
Customer	45,000,000
Supplier	3,000,000
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*

*RAID0 was used for database filegroups and tempdb, and RAID 0+1 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.*

A modified DBGEN version 2.3.0 was used for database population. The modified version differs only in column order for output flatfile. The only modified file, print.c, is included in Appendix F.1.

---

## 5.6. Database Load Time

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

The database load time was 07:25:45.

## 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 system configured 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 manufacturer 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.

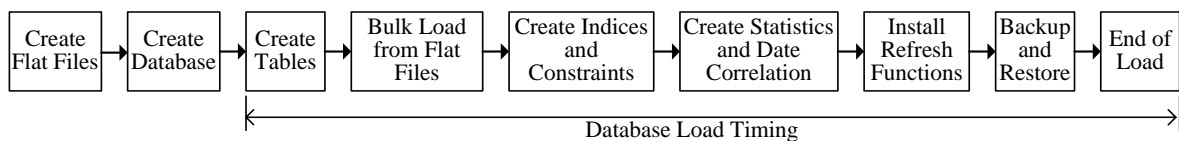
The Numerical Quantities summary contains the data storage ratio (34.80) for the system used.

## 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 2.3.0 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 4 logical devices
- 6 Rows were inserted into the database by running 8 concurrent threads, each of which performed a “bulk insert” operation that loaded one 16th 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. 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. The system was rebooted and SQL Server 2005 was started.
2. RF1 refresh transactions were run
3. Stream 0 execution was run
4. RF2 refresh transactions were run.

### 6.2. 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.

### 6.3. Number of Streams in the Throughput Test

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

Six streams were run for the throughput test.

### 6.4. 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 6.9.

### 6.5. 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 6.9.

### 6.6. 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 6.9.

#### 6.7. 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 6.9.

#### 6.8. 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 6.9.

#### 6.9. 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.*

	<b>QppH@300GB</b>	<b>QthH@300GB</b>	<b>QphH@300GB</b>
Run 1	14,799.0	8,979.0	11,527.4
Run 2	15,016.1	8,860.2	11,534.5

(Run1 was reported.)

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

## Numerical Quantities Summary

### Measurement Results

Scale Factor	300 GB
Total Data Storage/Database Size	34.80
Start of Database Load	4/19/06 16:35:33
End of Database Load	4/20/06 00:01:18
Database Load Time	07:25:45
Query Streams for Throughput Test	6
TPC-H Power	14,799.0
TPC-H Throughput	8,979.0
Composite Query per Hour Rating(QphH@300GB)	11,527.4
Total System Price Over 3 Years	\$810,428
TPC-H Price Performance Metric	\$70.30

### Measurement Intervals

Measurement Interval in Throughput Test(Ts)	15,877.0 seconds
---	------------------

### Duration of Stream Execution

	Seed	Query Start Date/Time		RF1 Start Date/Time		RF2 Start Date/Time		Query Duration
		Query End Date/Time		RF1 End Date/Time		RF2 End Date/Time		RF Duration
<b>Stream0</b>	420000118	04/20/2006	10:00:22	04/20/2006	09:58:48	04/20/2006	10:51:22	0:53:52
		04/20/2006	10:51:21	04/20/2006	10:00:22	04/20/2006	10:52:40	
<b>Stream1</b>	420000119	04/20/2006	10:52:41	04/20/2006	10:52:41	04/20/2006	14:57:48	3:55:55
		04/20/2006	14:48:36	04/20/2006	14:57:47	04/20/2006	14:59:11	
<b>Stream2</b>	420000120	04/20/2006	10:52:41	04/20/2006	14:59:13	04/20/2006	15:01:00	4:00:46
		04/20/2006	14:53:27	04/20/2006	15:01:00	04/20/2006	15:02:22	
<b>Stream3</b>	420000121	04/20/2006	10:52:41	04/20/2006	15:02:24	04/20/2006	15:04:12	3:54:38
		04/20/2006	14:47:19	04/20/2006	15:04:11	04/20/2006	15:05:44	
<b>Stream4</b>	420000122	04/20/2006	10:52:41	04/20/2006	15:05:47	04/20/2006	15:07:39	4:00:28
		04/20/2006	14:53:09	04/20/2006	15:07:38	04/20/2006	15:09:19	
<b>Stream5</b>	420000123	04/20/2006	10:52:42	04/20/2006	15:09:22	04/20/2006	15:11:16	3:51:26
		04/20/2006	14:44:08	04/20/2006	15:11:14	04/20/2006	15:13:04	
<b>Stream6</b>	420000124	04/20/2006	10:52:42	04/20/2006	15:13:08	04/20/2006	15:15:24	4:03:28
		04/20/2006	14:56:10	04/20/2006	15:15:22	04/20/2006	15:17:18	

**TPC-H Timing Intervals(in seconds):**

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8
Stream0	485.2	10.3	37.2	37.6	80.6	32.0	79.0	95.2
Stream1	3457.2	74.2	87.9	101.8	507.3	85.2	397.7	247.5
Stream2	3184.5	61.0	98.0	128.8	286.5	115.2	633.8	271.6
Stream3	3178.5	59.8	70.0	173.3	365.4	114.2	327.1	290.4
Stream4	3286.1	67.9	50.4	140.7	278.3	90.3	294.0	234.5
Stream5	3354.3	54.1	114.7	142.1	269.6	96.5	308.3	229.5
Stream6	3197.8	44.7	85.7	138.4	314.7	114.8	318.8	322.3
Min Qi	3178.5	44.7	50.4	101.8	269.6	85.2	294.0	229.5
Max Qi	3457.2	74.2	114.7	173.3	507.3	115.2	633.8	322.3
Avg Qi	3276.4	60.3	84.5	137.5	337.0	102.7	380.0	266.0
Query	Q9	Q10	Q11	Q12	Q13	Q14	Q15	Q16
Stream0	354.1	44.4	42.9	102.2	145.5	17.9	14.1	64.4
Stream1	1518.2	194.5	222.4	380.0	651.0	94.3	118.6	279.0
Stream2	1679.7	249.8	172.1	639.7	693.1	99.5	80.4	357.1
Stream3	2105.4	157.2	152.8	399.3	714.2	98.9	65.4	303.3
Stream4	1797.1	213.1	129.7	467.4	752.9	90.6	75.7	269.8
Stream5	1930.3	204.5	159.3	394.5	654.3	99.1	83.1	281.4
Stream6	2496.0	251.1	127.8	446.1	833.9	84.9	89.7	269.8
Min Qi	1518.2	157.2	127.8	380.0	651.0	84.9	65.4	269.8
Max Qi	2496.0	251.1	222.4	639.7	833.9	99.5	118.6	357.1
Avg Qi	1921.1	211.7	160.7	454.5	716.6	94.6	85.5	293.4
Query	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream0	44.2	447.0	91.7	25.2	709.6	99.2	94.1	77.9
Stream1	144.8	2448.6	241.3	135.9	2482.9	284.6	14706.3	83.2
Stream2	174.8	2614.3	303.6	140.8	2169.5	292.2	107.0	81.3
Stream3	156.9	2690.5	248.0	90.1	1996.4	320.7	106.9	92.8
Stream4	126.9	3009.5	296.7	162.4	2339.6	253.8	110.5	100.9
Stream5	101.9	2194.4	268.1	107.0	2481.9	356.9	112.1	108.6
Stream6	137.5	3127.4	280.0	151.8	1558.9	216.5	134.3	114.3
Min Qi	101.9	2194.4	241.3	90.1	1558.9	216.5	106.9	81.3
Max Qi	174.8	3127.4	303.6	162.4	2482.9	356.9	14706.3	114.3
Avg Qi	140.5	2680.8	273.0	131.3	2171.5	287.5	2546.2	96.9

6.10. System Activity Between Tests

*Any activity on the SUT that takes place between the conclusion of Run1 and the beginning of Run2 must be disclosed.*

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 48 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 total 72 RF2 threads and concurrently 24 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 six concurrent query streams. Each stream executes queries in the prescribed order for the appropriate Stream Id (01-06). Upon completion of each stream, a semaphore is set to indicate completion.
- Execute six 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, is 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 Configuration 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 following is an explanation to prove that the substitute disk drive (72GB FC 15K RPM) used in a priced configuration has the same performance as disk drives which are used in a benchmark tested configuration because 36GB FC 15K RPM used in a benchmark tested configuration will soon reach end of life. (Please refer to “8.3 Rules for Substitution of Components in a Priced Configuration” in TPC Pricing Specification Ver.1.0.1 for the rule of substitution component.  
[http://www.tpc.org/pricing/spec/Price\\_V1.0.1.pdf](http://www.tpc.org/pricing/spec/Price_V1.0.1.pdf))

Table 8.1 shows the specification of both disk drives: 36GB FC 15K RPM used for measurement in a benchmark tested configuration and 72GB FC 15K RPM used in a priced configuration. These data are extracted from “Cheetah 15K.3 FC Disc Drive ST373453FC/ST336753FC/ST318453FC Product Manual, Volume 1” published by Seagate.  
<http://www.seagate.com/support/disc/manuals/fc/100148129b.pdf>

Table 8.1 Disk Drive used in a benchmark tested configuration and a priced configuration

Item		Tested Conf. Disk	Priced Conf. Disk	
Hitachi Part Number		DF-F600-AEH36	DF-F600-AEH72	
Hitachi Product Name		36GB FC 15K RPM	72GB FC 15K RPM	
Model Number		ST336753FC	ST373453FC	
Product Family Name		Cheetah 15K.3	Cheetah 15K.3	
Capacity	Formatted 512 Kbytes/Sector	36.7GB	73.4GB	
Interface		2Gbps FC	2Gbps FC	
Performance	Internal transfer rate	609 to 891 Mb/s	609 to 891 Mb/s	
	Cache Size	8,192 Kbyte	8,192 Kbyte	
	Seek Time	Single Track(R/W)	0.2/0.4 msec	0.2/0.4 msec
		Full Stroke(R/W)	6.5/6.9 msec	6.5/6.9 msec
		Average(R/W)	3.6/3.9 msec	3.6/3.9 msec
	Average rotational latency		2.0 msec	2.0 msec
Disc rotation speed		15,000 rpm	15,000 rpm	
Configuration	Discs	2	4	
	Heads	4	8	
	Peak bits per inch	542 KBPI	542 KBPI	
	Tracks per inch	64,000 TPI	64,000 TPI	

According to table 8.1, 72GB FC 15K RPM used in a priced configuration has the equivalent performance to 36GB FC 15K RPM. Therefore we use 72GB FC 15K RPM in a priced configuration as a substituted component.

The detailed list of all hardware and software, including the three-year price, is listed in the Executive Summary at the beginning of this report. The price quotations are included in Appendix H, at the end of this report.

## 8.2. Three-Year Cost of System Configuration

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

The detailed list of all hardware and software, including the three-year price, is listed in the Executive Summary at the beginning of this report. The price quotations are included in Appendix H, at the end of this report.

## 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 is available at the time of publication.

The system software, Microsoft Windows Server 2003, Enterprise Edition for Itanium SP1 and database software, Microsoft SQL Server 2005 Enterprise Edition for Itanium SP1 are generally available at the time of publication.

## 8.4. Country-Specific Pricing

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

The configuration is priced for the United States of America.

## 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.*

This implementation of the TPC Benchmark H was audited by Francois Raab of InfoSizing, Inc. 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 SQL Server 2005 - 9.00.2040.00 (Intel IA-64)  
 Mar 13 2006 11:21:47  
 Copyright (c) 1988-2005 Microsoft Corporation  
 Enterprise Edition (64-bit) on Windows NT 5.2 (Build 3790: Service Pack 1)

**[Operating System Tuning]**

[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 properties]  
 The advanced performance settings was modified below from "System Properties".  
 - Visual Effects  
 The visuau effects setting was adjusted for best performance.  
 - Advanced  
 Both the processor scheduling and the memory usage were adjusted for best performance of programs.

**[System Information]**

System Information report written at: 04/20/06 18:15:39  
 System Name: EBS64  
 [System Summary]

ItemValue  
 OS NameMicrosoft(R) Windows(R) Server 2003, Enterprise Edition for 64-Bit Itanium-based Systems  
 Version5.2.3790 Service Pack 1 Build 3790  
 Other OS Description Not Available  
 OS ManufacturerMicrosoft Corporation  
 System NameEBS64  
 System Manufacturerhitachi  
 System Modelserver BladeSymphony  
 System TypeItanium (TM) -based System  
 Processoria64 Family 31 Model 2 Stepping 2 GenuineIntel ~1667 Mhz  
 Processoria64 Family 31 Model 2 Stepping 2 GenuineIntel ~1667 Mhz  
 Processoria64 Family 31 Model 2 Stepping 2 GenuineIntel ~1667 Mhz  
 Processoria64 Family 31 Model 2 Stepping 2 GenuineIntel ~1667 Mhz  
 Processoria64 Family 31 Model 2 Stepping 2 GenuineIntel ~1667 Mhz  
 Processoria64 Family 31 Model 2 Stepping 2 GenuineIntel ~1667 Mhz  
 Processoria64 Family 31 Model 2 Stepping 2 GenuineIntel ~1667 Mhz  
 Processoria64 Family 31 Model 2 Stepping 2 GenuineIntel ~1667 Mhz  
 BIOS Version/DateHITACHI 50-27, 10/27/2005

SMBIOS Version2.3  
 Windows DirectoryC:\WINDOWS  
 System DirectoryC:\WINDOWS\system32  
 Boot Device¥Device¥HarddiskVolume1  
 LocaleUnited States  
 Hardware Abstraction LayerVersion = "5.2.3790.1830 (srv03\_sp1\_rtm.050324-1447)"  
 User NameNot Available  
 Time ZoneTokyo Standard Time  
 Total Physical Memory130,996.68 MB  
 Available Physical Memory126.35 GB  
 Total Virtual Memory128.60 GB  
 Available Virtual Memory128.28 GB  
 Page File Space2.00 GB  
 Page File C:\pagefile.sys

**[DBMS Startup Parameters]**

Microsoft(R) SQL ServerTM 2005 Startup Parameters  
 sqlservr -c -x -E -T834 -T2301  
 -c  
 Start SQL Server 2005 independently of the Windows NT Service Control Manager  
 -x  
 Disable the keeping of CPU time and cache-hit ratio statistics  
 -E  
 Increase the number of consecutive extents allocated to 4  
 -T834  
 Force SQL Server to use large pages  
 -T2301  
 The query plan optimizer to provide a more aggressive plan for query execution

**[DBMS Configuration Parameters]**

Microsoft(R) SQL ServerTM 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	255	255
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	0	0
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	126000	126000
max text repl size (B)	0	2147483647	65536	65536
max worker threads	128	32767	0	0
media retention	0	365	0	0
min memory per query (KB)	512	2147483647	512	512
min server memory (MB)	0	2147483647	116000	116000
nested triggers	0	1	1	1
network packet size (B)	512	32767	32767	32767
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	1	1
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	600	600
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

### 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="E:\%DBNAME%\_root.mdf", SIZE=16MB, FILEGROWTH=1MB),
FILEGROUP LINEITEM_FG
(NAME=%DBNAME%_li_0, FILENAME="E:\mnt\li\0", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_1, FILENAME="E:\mnt\li\1", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_2, FILENAME="E:\mnt\li\2", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_3, FILENAME="E:\mnt\li\3", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_4, FILENAME="E:\mnt\li\4", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_5, FILENAME="E:\mnt\li\5", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_6, FILENAME="E:\mnt\li\6", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_7, FILENAME="E:\mnt\li\7", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_8, FILENAME="E:\mnt\li\8", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_9, FILENAME="E:\mnt\li\9", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_10, FILENAME="E:\mnt\li\10", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_11, FILENAME="E:\mnt\li\11", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_12, FILENAME="E:\mnt\li\12", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_13, FILENAME="E:\mnt\li\13", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_li_14, FILENAME="E:\mnt\li\14", SIZE=40GB, FILEGROWTH=0),
FILEGROUP GENERAL_FG
(NAME=%DBNAME%_gen_0, FILENAME="E:\mnt\ge\0", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_1, FILENAME="E:\mnt\ge\1", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_2, FILENAME="E:\mnt\ge\2", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_3, FILENAME="E:\mnt\ge\3", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_4, FILENAME="E:\mnt\ge\4", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_5, FILENAME="E:\mnt\ge\5", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_6, FILENAME="E:\mnt\ge\6", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_7, FILENAME="E:\mnt\ge\7", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_8, FILENAME="E:\mnt\ge\8", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_9, FILENAME="E:\mnt\ge\9", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_10, FILENAME="E:\mnt\ge\10", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_11, FILENAME="E:\mnt\ge\11", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_12, FILENAME="E:\mnt\ge\12", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_13, FILENAME="E:\mnt\ge\13", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_gen_14, FILENAME="E:\mnt\ge\14", SIZE=30GB, FILEGROWTH=0),
FILEGROUP INDEX_FG
```

```
(NAME=%DBNAME%_idx_0, FILENAME="E:\mnt\idx\0", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_1, FILENAME="E:\mnt\idx\1", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_2, FILENAME="E:\mnt\idx\2", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_3, FILENAME="E:\mnt\idx\3", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_4, FILENAME="E:\mnt\idx\4", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_5, FILENAME="E:\mnt\idx\5", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_6, FILENAME="E:\mnt\idx\6", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_7, FILENAME="E:\mnt\idx\7", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_8, FILENAME="E:\mnt\idx\8", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_9, FILENAME="E:\mnt\idx\9", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_10, FILENAME="E:\mnt\idx\10", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_11, FILENAME="E:\mnt\idx\11", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_12, FILENAME="E:\mnt\idx\12", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_13, FILENAME="E:\mnt\idx\13", SIZE=30GB, FILEGROWTH=0),
(NAME=%DBNAME%_idx_14, FILENAME="E:\mnt\idx\14", SIZE=30GB, FILEGROWTH=0),
FILEGROUP LOAD_FG
(NAME=%DBNAME%_load_0, FILENAME="E:\mnt\load\0", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_1, FILENAME="E:\mnt\load\1", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_2, FILENAME="E:\mnt\load\2", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_3, FILENAME="E:\mnt\load\3", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_4, FILENAME="E:\mnt\load\4", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_5, FILENAME="E:\mnt\load\5", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_6, FILENAME="E:\mnt\load\6", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_7, FILENAME="E:\mnt\load\7", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_8, FILENAME="E:\mnt\load\8", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_9, FILENAME="E:\mnt\load\9", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_10, FILENAME="E:\mnt\load\10", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_11, FILENAME="E:\mnt\load\11", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_12, FILENAME="E:\mnt\load\12", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_13, FILENAME="E:\mnt\load\13", SIZE=40GB, FILEGROWTH=0),
(NAME=%DBNAME%_load_14, FILENAME="E:\mnt\load\14", SIZE=40GB, FILEGROWTH=0)
LOG ON
(NAME=%DBNAME%_log_0, FILENAME="E:\mnt\log\0", SIZE=100GB, FILEGROWTH=0)
```

### 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_TYPE varchar(25) not null,
 P_SIZE int not null,
 P_BRAND char(10) not null,
 P_NAME varchar(55) not null,
 P_CONTAINER char(10) not null,
 P_MFGR char(25) not null,
 P_RETAILPRICE float not null,
 P_COMMENT varchar(23) not null)
on LOAD_FG
```

```

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

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

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

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

create table LINEITEM
(L_SHIPDATE datetime not null,
 L_ORDERKEY bigint not null,
 L_DISCOUNT float not null,
 L_EXTENDEDPRICE float not null,
 L_SUPPKEY int not null,
 L_QUANTITY float not null,
 L_RETURNFLAG char(1) not null,
 L_PARTKEY int not null,
 L_LINESTATUS char(1) not null,
 L_TAX float not null,
 L_COMMITDATE datetime not null,

```

```

L_RECEIPTDATE datetime not null,
L_SHIPMODE char(10) not null,
L_LINENUMBER int not null,
L_SHIPINSTRUCT char(25) not null,
L_COMMENT varchar(44) not null)
on LOAD_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 LOAD_FG

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

```

### CreateIndexes\_1.sql

```

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

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

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

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

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

create nonclustered index C_NATIONKEY_IDX on CUSTOMER(C_NATIONKEY)
with (FILLFACTOR=100,SORT_IN_TEMPDB=ON) on INDEX_FG

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

```

```
create index PS_SUPPKEY_IDX on PARTSUPP(PS_SUPPKEY) include(PS_PARTKEY)
with (FILLFACTOR=100, SORT_IN_TEMPDB=ON) on INDEX_FG
```

```
create clustered index O_ORDERDATE_CLIDX on ORDERS(O_ORDERDATE)
with (PAD_INDEX=ON, FILLFACTOR=95, SORT_IN_TEMPDB=ON) on GENERAL_FG
```

```
alter table ORDERS add constraint PK_O_ORDERKEY primary key (O_ORDERKEY)
with (PAD_INDEX=ON, FILLFACTOR=95, SORT_IN_TEMPDB=ON) on GENERAL_FG
```

```
create nonclustered index O_CUSTKEY_IDX on ORDERS(O_CUSTKEY)
with (PAD_INDEX=ON, FILLFACTOR=95, SORT_IN_TEMPDB=ON) on INDEX_FG
```

### CreateIndexes\_2.sql

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

```
create index L_ORDERKEY_IDX on LINEITEM(L_ORDERKEY)
with (PAD_INDEX=ON, FILLFACTOR=95, SORT_IN_TEMPDB=ON) on INDEX_FG
```

```
create index L_PARTKEY_IDX on LINEITEM(L_PARTKEY)
with (PAD_INDEX=ON, FILLFACTOR=95, SORT_IN_TEMPDB=ON) on INDEX_FG
```

### CreateFK.sql

```
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)
```

### CreateBackupDevices.sql

```
-- File: CREATEBACKUPDEVICES.SQL
-- Microsoft TPC-H Benchmark Kit Ver. 2.00
-- Copyright Microsoft, 1999
--
```

```
if exists (select name from sysdevices where name = 'BackupDev1')
exec sp_dropdevice BackupDev1
GO
```

```
if exists (select name from sysdevices where name = 'BackupDev2')
exec sp_dropdevice BackupDev2
GO
```

```
if exists (select name from sysdevices where name = 'BackupDev3')
exec sp_dropdevice BackupDev3
GO
```

```
if exists (select name from sysdevices where name = 'BackupDev4')
exec sp_dropdevice BackupDev4
GO
```

```
sp_addumpdevice 'disk', 'BackupDev1', 'E:\mnt\%backup%\0\TPCHbackup1_300G'
GO
```

```
sp_addumpdevice 'disk', 'BackupDev2', 'E:\mnt\%backup%\1\TPCHbackup2_300G'
GO
```

```
sp_addumpdevice 'disk', 'BackupDev3', 'E:\mnt\%backup%\2\TPCHbackup3_300G'
GO
```

```
sp_addumpdevice 'disk', 'BackupDev4', 'E:\mnt\%backup%\3\TPCHbackup4_300G'
GO
```

### BackupDatabase.sql

```
--File: BACKUPDATABASE.SQL
-- Microsoft TPC-H Benchmark Kit Version 2.00
-- Copyright Microsoft, 1999
--
```

```
backup database %DBNAME% to BackupDev1, BackupDev2, BackupDev3, BackupDev4 with
init, stats=10
```



**APPENDIX C : QUERY TEXT & OUTPUT**

**QUERY 1 FOR VALIDATION**

-----  
 -- using default substitutions

/\* TPC\_H Query 1 - Pricing Summary Report \*/

```

SELECT L_RETURNFLAG,
       L_LINESTATUS,
       SUM(L_QUANTITY) AS SUM_QTY,
       SUM(L_EXTENDEDPRI) AS SUM_BASE_PRICE,
       SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) AS SUM_DISC_PRICE,
       SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)*(1+L_TAX)) AS SUM_CHARGE,
       AVG(L_QUANTITY) AS AVG_QTY,
       AVG(L_EXTENDEDPRI) AS AVG_PRICE,
       AVG(L_DISCOUNT) AS AVG_DISC,
       COUNT_BIG(*) AS COUNT_ORDER
FROM LINEITEM
WHERE L_SHIPDATE <= dateadd(dd, -90, '1998-12-01')
GROUP BY L_RETURNFLAG,
         L_LINESTATUS
ORDER BY L_RETURNFLAG,
         L_LINESTATUS
  
```

L_RETURNFLAG	L_LINESTATUS	SUM_QTY	SUM_BASE_PRICE
SUM_DISC_PRICE	SUM_CHARGE	AVG_QTY	
AVG_PRICE	AVG_DISC	COUNT_ORDER	
A	F	37734107.000000	56586554400.730209
53758257134.870033	55909065222.827194	25.522006	
38273.129735	0.049985	1478493	
N	F	991417.000000	1487504710.379998
1413082168.054106	1469649223.194380	25.516472	
38284.467761	0.050093	38854	
N	O	74476040.000000	111701729697.740050
106118230307.604520	110367043872.497220	25.502227	
38249.117989	0.049997	2920374	
R	F	37719753.000000	56568041380.899704
53741292684.604240	55889619119.832001	25.505794	
38250.854626	0.050009	1478870	

(4 row(s) affected)

**QUERY 2 FOR VALIDATION**

-----  
 -- using default substitutions

/\* 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 = 15 AND
      P_TYPE LIKE '%BRASS' 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
  
```

S_ACCTBAL	P_PARTKEY	P_MFGR	S_NAME	S_ADDRESS	N_NAME
9938.530000			Supplier#000005359		UNITED KINGDOM
185358			Manufacturer#4	QKuHYh,vZGiwu2FWEJoLDx04	
33-429-790-6131			blithely silent pinto beans are furiously. slyly final deposits		
acros					
9937.840000			Supplier#000005969		ROMANIA
108438			Manufacturer#1	ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa	
29-520-692-3537			carefully slow deposits use furiously. slyly ironic platelets		

above the ironic				32-139-873-8571 furiously even deposits affix thinly special theodolites.
9936.220000	Supplier#000005250	UNITED KINGDOM		furio
249	Manufacturer#4	B3rqp0xbSEim4Mpy2RH J		9643.550000
33-320-228-2957 blithely special packages are. stealthily express deposits				Supplier#000005148
across the closely final instructi				ROMANIA
9923.770000	Supplier#000002324	GERMANY		107617
29821	Manufacturer#4	y3OD9UywSTOk		Manufacturer#1
17-779-299-1839 quickly express packages breach quiet pinto beans. requ				kt4ciVfSlx9z4s79p Js825
9871.220000	Supplier#000006373	GERMANY		29-252-617-4850 doggedly even ideas boost furiously against the furiously
43868	Manufacturer#5	J8fcXWstqM		express
17-813-485-8637 never silent deposits integrate furiously blit				9624.820000
9870.780000	Supplier#000001286	GERMANY		Supplier#000001816
81285	Manufacturer#2	YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH		FRANCE
17-516-924-4574 final theodolites cajole slyly special,				34306
9870.780000	Supplier#000001286	GERMANY		Manufacturer#3
181285	Manufacturer#4	YKA,E2fjiVd7eUrzp2Ef8j1QxGo2DFnosaTEH		e7vab91vLJPWxxZnewmnDBpDmxYHrb
17-516-924-4574 final theodolites cajole slyly special,				16-392-237-6726 blithely regular accounts cajole furiously. regular
9852.520000	Supplier#000008973	RUSSIA		9624.780000
18972	Manufacturer#2	t5L67YdBYYH6o,Vz24jpdYQ9		Supplier#000009658
32-188-594-7038 quickly regular instructions wake-- carefully unusual braids				ROMANIA
into the expres				189657
9847.830000	Supplier#000008097	RUSSIA		Manufacturer#1
130557	Manufacturer#2	xMe97bpE69NzdWLoX		oE9uBgEfSS4opIcepXyAYM,x
32-375-640-3593 slyly regular dependencies sleep slyly furiously express dep				29-748-876-2014 regular deposits haggle. furiously express asympto
9847.570000	Supplier#000006345	FRANCE		9612.940000
86344	Manufacturer#1	VSt3rzK3qG698u6ld8HhOByvrTcSTsvQ1DQDag		Supplier#000003228
16-886-766-7945 silent pinto beans should have to snooze carefully along the				ROMANIA
final reques				120715
9847.570000	Supplier#000006345	FRANCE		Manufacturer#2
173827	Manufacturer#2	VSt3rzK3qG698u6ld8HhOByvrTcSTsvQ1DQDag		KDdpNKN3cWu7ZSrbdq7AfSLxx,qWB
16-886-766-7945 silent pinto beans should have to snooze carefully along the				29-325-784-8187 carefully pending accounts serve. furiously close deposits
final reques				boost slyly. q
9836.930000	Supplier#000007342	RUSSIA		9612.940000
4841	Manufacturer#4	JOLK7C1,7xrEZSSOw		Supplier#000003228
32-399-414-5385 final accounts haggle. bold accounts are furiously dugouts.				ROMANIA
furiously silent asymptotes are slyly				198189
9817.100000	Supplier#000002352	RUSSIA		Manufacturer#4
124815	Manufacturer#2	4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw		KDdpNKN3cWu7ZSrbdq7AfSLxx,qWB
32-551-831-1437 blithely pending packages across the ironic accounts grow slyly				29-325-784-8187 carefully pending accounts serve. furiously close deposits
after the furiously				boost slyly. q
9817.100000	Supplier#000002352	RUSSIA		9571.830000
152351	Manufacturer#3	4LfoHUZjgjEbAKw TgdKcgOc4D4uCYw		Supplier#000004305
32-551-831-1437 blithely pending packages across the ironic accounts grow slyly				ROMANIA
after the furiously				179270
9739.860000	Supplier#000003384	FRANCE		Manufacturer#2
138357	Manufacturer#2	o,Z3v4POifevE k9U1b 6J1ucX,I		qNHZ7WmCzygwMPRDO9Ps
16-494-913-5925 slyly ironic theodolites hag				29-973-481-1831 furiously final deposits
9721.950000	Supplier#000008757	UNITED KINGDOM		9558.100000
156241	Manufacturer#3	Atg6GnM4d2T		Supplier#000003532
33-821-407-2995 ironic, even dolphins above the furiously ironic foxes sleep				UNITED KINGDOM
slyly around the caref				88515
9681.330000	Supplier#000008406	RUSSIA		Manufacturer#4
78405	Manufacturer#1	,qUuXcftUl		EOeuiiOn210VpTlGguufFDFsbN1p0lhpxHp
				33-152-301-2164 daring, sly accounts breach about th
				9492.790000
				Supplier#000005975
				GERMANY
				25974
				Manufacturer#5
				S6mIiCTx82z7lV
				17-992-579-4839 always pending packages boost slyly.
				9461.050000
				Supplier#000002536
				UNITED KINGDOM
				20033
				Manufacturer#1
				8mmGbyzaU 7ZS2wJumTibypncu9pNkDc4FYA
				33-556-973-5522 even foxes are quickly furiously express requests. packages
				9453.010000
				Supplier#000000802
				ROMANIA
				175767
				Manufacturer#1
				,6HYXB4uaHITmtMBj4Ak57Pd
				29-342-882-6463 final, regular packages across the slowly regular packag
				9408.650000
				Supplier#000007772
				UNITED KINGDOM
				117771
				Manufacturer#4
				AiC5YAH,gdu0i7
				33-152-491-1126 blithely final ideas sleep carefully. requests are
				9359.610000
				Supplier#000004856
				ROMANIA
				62349
				Manufacturer#5
				HYogcF3Jb yhl
				29-334-870-9731 carefully unusual packages sleep carefully even ideas. dogged
				account
				9357.450000
				Supplier#000006188
				UNITED KINGDOM
				138648
				Manufacturer#1
				g801,ssP8wpTk4Hm
				33-583-607-1633 carefully regular deposits wake carefully furiously even i
				9352.040000
				Supplier#000003439
				GERMANY
				170921
				Manufacturer#4
				qYPDgoiBGhCYxjgC
				17-128-996-4650 fluffily regular pinto beans wake. unusual, final ideas c
				9312.970000
				Supplier#000007807
				RUSSIA
				90279
				Manufacturer#5
				oGYMPck9XHGB2PBfKRnHA
				32-673-872-5854 unusual asymptotes above the
				9312.970000
				Supplier#000007807
				RUSSIA
				100276
				Manufacturer#5
				oGYMPck9XHGB2PBfKRnHA

32-673-872-5854	unusual asymptotes above the		
9280.270000	Supplier#000007194	ROMANIA	
47193	Manufacturer#3	zhRUQkBSrFYxIAXTflnj vyGRQjeK	
29-318-454-2133	slyly ironic requests despite the unusual ins		
9274.800000	Supplier#000008854	RUSSIA	
76346	Manufacturer#3	1xhLoOUM7I3mZ1mKnerw OSqdbb4QbGa	
32-524-148-5221	ruthlessly ironic instructions along the regular, furious		
requests integrate car			
9249.350000	Supplier#000003973	FRANCE	
26466	Manufacturer#1	d18GiDsL6Wm2IsGXM, RZf1jCsgZAOjNYVThTRP4	
16-722-866-1658	quickly ironic sauternes use b		
9249.350000	Supplier#000003973	FRANCE	
33972	Manufacturer#1	d18GiDsL6Wm2IsGXM, RZf1jCsgZAOjNYVThTRP4	
16-722-866-1658	quickly ironic sauternes use b		
9208.700000	Supplier#000007769	ROMANIA	
40256	Manufacturer#5	rsimdze 5o9P Ht7xS	
29-964-424-9649	furiously ruthless epitaphs among the furiously regular		
accounts use slowly fluffily ev			
9201.470000	Supplier#000009690	UNITED KINGDOM	
67183	Manufacturer#5	CB BnUTlmi5zdeEl7R7	
33-121-267-9529	blithely unusual accounts integrate slyly. platelets		
9192.100000	Supplier#000000115	UNITED KINGDOM	
85098	Manufacturer#3	nJ 2t0f7Ve, wL1, 6WzGBJLNUCK1sV	
33-597-248-1220	slyly bold pinto beans boost across the furiously regular		
packages. carefully regu			
9189.980000	Supplier#000001226	GERMANY	
21225	Manufacturer#4	qsLCqSvLyZfuXipjz	
17-725-903-1381	final, express instruction		
9128.970000	Supplier#000004311	RUSSIA	
146768	Manufacturer#5	I8IjnXd7NSJRs594RxsRR0	
32-155-440-7120	regular pinto beans sleep ca		
9104.830000	Supplier#000008520	GERMANY	
150974	Manufacturer#4	RqRVDgD0ER J9 b41vR2, 3	
17-728-804-1793	deposits sleep carefully e		
9101.000000	Supplier#000005791	ROMANIA	
128254	Manufacturer#5	zub2zCV, jhHPPQqi, P2INAjElzI n66cOEOXFG	
29-549-251-5384	carefully ironic packages after the		
9094.570000	Supplier#000004582	RUSSIA	
39575	Manufacturer#1	WB0XkCSG3r, mnQ n, h9VIxj9ARHFvKgMdf	
32-587-577-1351	asymptotes above the slyly even requests haggle furiously about		
the regular accounts			
8996.870000	Supplier#000004702	FRANCE	
102191	Manufacturer#5	8XVcQK23akp	
16-811-269-8946	stealthy requests haggle c		
8996.140000	Supplier#000009814	ROMANIA	
139813	Manufacturer#2	af005pg831PU4IDVmEylXZVqYZQzSDLYLAmR	
29-995-571-8781	ironic theodolites are evenly unusual requests-- pending pinto		
beans across the in			
8968.420000	Supplier#000010000	ROMANIA	
119999	Manufacturer#5	aTGLEusCiL4F PDBdv665XBjPyCOB0i	
29-578-432-2146	furiously final ideas believe furiously. furiously final ideas		
8936.820000	Supplier#000007043	UNITED KINGDOM	
109512	Manufacturer#1	FVajceZInZdbJE6Z9XsRUxrUEpiwHDrOXi, 1Rz	
33-784-177-8208	furiously regular excuses wake after the blithely special pinto		
beans? even instructions sl			
8929.420000	Supplier#000008770	FRANCE	
173735	Manufacturer#4	R7cG26TtXrHAP9 HckhfrI	
16-242-746-9248	final accounts sleep furiously. blithely ironic foxes wake		
boldly across the furiously s			
8920.590000	Supplier#000003967	ROMANIA	
26460	Manufacturer#1	eHoAXe62SY9	
29-194-731-3944	quickly even requests should have to affix blithely-- fur		
8920.590000	Supplier#000003967	ROMANIA	
173966	Manufacturer#2	eHoAXe62SY9	
29-194-731-3944	quickly even requests should have to affix blithely-- fur		
8913.960000	Supplier#000004603	UNITED KINGDOM	
137063	Manufacturer#2	OUzlvMUr7n, utLxmPNeYKSF3T24OXskxB5	
33-789-255-7342	slyly ironic packages detect furious accounts. ironic de		
8877.820000	Supplier#000007967	FRANCE	
167966	Manufacturer#5	A3pilBARM4nx6R, qrwFoRPU	
16-442-147-9345	final deposits after the silent deposits ha		
8862.240000	Supplier#000003323	ROMANIA	
73322	Manufacturer#3	W9 lYcsC9FwBqk3ItL	
29-736-951-3710	unusual, pending theodolites integrate furiously slyly even		
pinto beans. unusual sheaves sleep befor			
8841.590000	Supplier#000005750	ROMANIA	
100729	Manufacturer#5	Erx3lAgu0g62iaHF9x50uMH4EgeN9hEG	
29-344-502-5481	excuses after the blithely regular packages mold carefully		
deposits. regular a			
8781.710000	Supplier#000003121	ROMANIA	
13120	Manufacturer#5	wNqTogx238ZYCamFb, 50v, bj 4IbNFW9Bvw1xP	
29-707-291-5144	packages are quickly after the final, even packages. furiously		
regular			
8754.240000	Supplier#000009407	UNITED KINGDOM	
179406	Manufacturer#4	CHRCbkaWcf5B	
33-903-970-9604	regular dependencies haggle across the carefully bold		
8691.060000	Supplier#000004429	UNITED KINGDOM	
126892	Manufacturer#2	k, BQms5UhoAF1B2Asi, fLib	
33-964-337-5038	quickly special foxes against the furiously silent platelets		
wake quickly after t			
8655.990000	Supplier#000006330	RUSSIA	
193810	Manufacturer#2	UozlaENr0ytKe2w6CeIEWFwn iO3S8Rae7Ou	
32-561-198-3705	blithely even packages alongside		
8638.360000	Supplier#000002920	RUSSIA	
75398	Manufacturer#1	Je2a8bszf3L	
32-122-621-7549	express deposits wake. furiously silent requests wake carefully		
silent instru			
8638.360000	Supplier#000002920	RUSSIA	
170402	Manufacturer#3	Je2a8bszf3L	
32-122-621-7549	express deposits wake. furiously silent requests wake carefully		
silent instru			
8607.690000	Supplier#000006003	UNITED KINGDOM	
76002	Manufacturer#2	EH9wADcEiuenMONR08zDwMidw, 52Y2RyILEiA	
33-416-807-5206	always special foxes wake slyly bold, ironic accounts. ironic		
instructions affix carefull			
8569.520000	Supplier#000005936	RUSSIA	

5935	Manufacturer#5	jXaNz6vwnEWJ2ksLZJpjtgt0bY2a3AU	8328.460000	Supplier#000001744	ROMANIA
32-644-251-7916		packages sleep furiously. special requests about the fluffily	69237	Manufacturer#5	oLo3fV64q2,FKHa3p,qHnS7Yzv,ps8
		even accounts detect	29-330-728-5873		blithely silent excuses are slyly above the furiously even
8564.120000	Supplier#000000033	GERMANY	courts		
110032	Manufacturer#1	gfeKpYw3400LOSDyWXA6YalQmq1w6YB9f3R	8307.930000	Supplier#000003142	GERMANY
17-138-897-9374		ironic instructions are. special pearls above	18139	Manufacturer#1	dqblvV8dCNAorGlJ
8553.820000	Supplier#000003979	ROMANIA	17-595-447-6026		theodolites sleep blithely carefully regular warhorses. slyly
143978	Manufacturer#4	BfmVhCAnCMY3jzpjUMy4CNWs9 HzpdQR7INJU	regular ins		
29-124-646-4897		express, ironic pinto beans cajole around the express, even	8231.610000	Supplier#000009558	RUSSIA
packages. qu			192000	Manufacturer#2	mcdgen,yT1iJDHDS5fV
8517.230000	Supplier#000009529	RUSSIA	32-762-137-5858		slyly regular theodolites sleep fluffily express depos
37025	Manufacturer#5	e44R8o7JAIS9iMcr	8152.610000	Supplier#000002731	ROMANIA
32-565-297-8775		furiously silent requests cajole furiously furiously ironic	15227	Manufacturer#4	nluXJCuY1tu
foxes. slyly express p			29-805-463-2030		gifts use. slyly silent ideas are carefully beneath the silent
8517.230000	Supplier#000009529	RUSSIA	instructions. slyly sil		
59528	Manufacturer#2	e44R8o7JAIS9iMcr	8109.090000	Supplier#000009186	FRANCE
32-565-297-8775		furiously silent requests cajole furiously furiously ironic	99185	Manufacturer#1	wgfosrVPexl9pEXWyaqlBMDYYf
foxes. slyly express p			16-668-570-1402		quickly pending requests are blithely along the ironic, final
8503.700000	Supplier#000006830	RUSSIA	requests; instr		
44325	Manufacturer#4	BC4WFCYRUZyaIgcH U 4S	8102.620000	Supplier#000003347	UNITED KINGDOM
32-147-878-5069		quickly regular excuses detect evenly around	18344	Manufacturer#5	m CtXS2S16i
8457.090000	Supplier#000009456	UNITED KINGDOM	33-454-274-8532		packages grow special orbits. regular theodolites about the
19455	Manufacturer#1	7SBhZs8gP1cJtT0Qf433YBk	carefully pe		
33-858-440-4349		carefully final accounts sleep blithely special foxes. slyly	8046.070000	Supplier#000008780	FRANCE
regular pinto beans alon			191222	Manufacturer#3	AczzuE0UK9osj ,Lx0Jmh
8441.400000	Supplier#000003817	FRANCE	16-473-215-6395		regular epitaphs integrate slyly.
141302	Manufacturer#2	hU3fz3xL78	8042.090000	Supplier#000003245	RUSSIA
16-339-356-5115		blithely blithe ideas are	135705	Manufacturer#4	Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y
8432.890000	Supplier#000003990	RUSSIA	32-836-132-8872		carefully regular instructions integrate blithely silent foxes.
191470	Manufacturer#1	wehBBp1RQbfxAYDASS75MsywmsKHRVdkrvNe6m	furiously express instructions haggl		
32-839-509-9301		final requests along the blithely ironic packages kindle	8042.090000	Supplier#000003245	RUSSIA
against the carefully fina			150729	Manufacturer#1	Dh8Ikg39onrbOL4DyTfGw8a9oKUX3d9Y
8431.400000	Supplier#000002675	ROMANIA	32-836-132-8872		carefully regular instructions integrate blithely silent foxes.
5174	Manufacturer#1	HJFStOu9R5NGPOegKhgbzBdyvrG2yh8w	furiously express instructions haggl		
29-474-643-1443		express, final deposits cajole carefully. stealthily unusual	7992.400000	Supplier#000006108	FRANCE
requests			118574	Manufacturer#1	8tBydnTDwUqfBfFV413
8407.040000	Supplier#000005406	RUSSIA	16-974-998-8937		regular pinto beans are after
162889	Manufacturer#4	j7 gYF5RW8DC5UurjKC	7980.650000	Supplier#000001288	FRANCE
32-626-152-4621		quickly final sheaves boost. car	13784	Manufacturer#4	zE,7HgVPrCn
8386.080000	Supplier#000008518	FRANCE	16-646-464-8247		unusual pinto beans cajole furiously according t
36014	Manufacturer#3	2jqzqqAVe9c9rMVGP,n9nTsQXulNLTUYoJjEDcqWV	7950.370000	Supplier#000008101	GERMANY
16-618-780-7481		slyly ironic theodolites are slyly. dogged, pendin	33094	Manufacturer#5	kkYvL6IuvojJgTNG IKkaXQDYgx8ILohj
8376.520000	Supplier#000005306	UNITED KINGDOM	17-627-663-8014		quickly regular requests are furiously. pending deposits wake
190267	Manufacturer#5	9t8Y8 QqSIs0ADPt6NLdk,TP5zyRx41oBulgoGc9	7937.930000	Supplier#000009012	ROMANIA
33-632-514-7931		furiously even instructions integrate during the furiously	83995	Manufacturer#2	iUiTziH,Ek3i4lwSgunXMgrcTzwdb
regular re			29-250-925-9690		blithely bold ideas haggle quickly final, regular request
8348.740000	Supplier#000008851	FRANCE	7914.450000	Supplier#000001013	RUSSIA
66344	Manufacturer#4	nWxi7GwEbhw1	125988	Manufacturer#2	riRcntps4KEDtYScjpmIWeYf6mNnr
16-796-240-2472		ironic instructions nag slyly against the slyly even	32-194-698-3365		final, ironic theodolites alongside of the ironic
theodolites. requests alongside of			7912.910000	Supplier#000004211	GERMANY
8338.580000	Supplier#000007269	FRANCE	159180	Manufacturer#5	2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG
17268	Manufacturer#4	ZwhJSwABUoiB04,3	17-266-947-7315		final requests integrate slyly above the silent, even
16-267-277-4365		ruthlessly regular asymptotes a	7912.910000	Supplier#000004211	GERMANY

```

184210      Manufacturer#4          2wQRVovHrm3,v03IKzfTd,1PYsFXQFFOG
17-266-947-7315 final requests integrate slyly above the silent, even
7894.560000      Supplier#000007981          GERMANY
85472      Manufacturer#4          NSJ96vMROAbeXP
17-963-404-3760 regular, even theodolites integrate carefully. bold, special
theodolites are slyly fluffily iron
7887.080000      Supplier#000009792          GERMANY
164759      Manufacturer#3          Y28ITVeYrit3kIGdV2K8fsZ V2UqT5H10tz
17-988-938-4296 pending, ironic packages sleep among the carefully ironic
accounts. quickly final accounts
7871.500000      Supplier#000007206          RUSSIA
104695      Manufacturer#1          3w fNCnrVmvJjE95sgWZzvW
32-432-452-7731 furiously dogged pinto beans cajole. bold, express notornis
until the slyly pending
7852.450000      Supplier#000005864          RUSSIA
8363      Manufacturer#4          WCNfBPZeSXh3h,c
32-454-883-3821 blithely regular deposits
7850.660000      Supplier#000001518          UNITED KINGDOM
86501      Manufacturer#1          ONda3YJiHKJOC
33-730-383-3892 furiously final accounts wake carefully idle requests. even
dolphins wake acc
7843.520000      Supplier#000006683          FRANCE
11680      Manufacturer#4          2Z0JGkiv01Y00oCFwUGfviIbhzCdy
16-464-517-8943 carefully bold accounts doub

```

(100 row(s) affected)

**QUERY 3 FOR VALIDATION**

```

-----
-- using default substitutions

/* TPC_H Query 3 - Shipping Priority */

```

```

SELECT TOP 10
L_ORDERKEY,
SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) AS REVENUE,
O_ORDERDATE,
O_SHIPRIORITY
FROM CUSTOMER,
ORDERS,
LINEITEM
WHERE C_MKTSEGMENT = 'BUILDING' AND
C_CUSTKEY = O_CUSTKEY AND
L_ORDERKEY = O_ORDERKEY AND
O_ORDERDATE < '1995-03-15' AND
L_SHIPDATE > '1995-03-15'
GROUP BY L_ORDERKEY,
O_ORDERDATE,
O_SHIPRIORITY
ORDER BY REVENUE DESC,
O_ORDERDATE

```

L_ORDERKEY	REVENUE	O_ORDERDATE
------------	---------	-------------

**O\_SHIPRIORITY**

```

-----
-----
2456423      406181.011100      1995-03-05 00:00:00.000 0
3459808      405838.698900      1995-03-04 00:00:00.000 0
492164      390324.061000      1995-02-19 00:00:00.000 0
1188320      384537.935900      1995-03-09 00:00:00.000 0
2435712      378673.055800      1995-02-26 00:00:00.000 0
4878020      378376.795200      1995-03-12 00:00:00.000 0
5521732      375153.921500      1995-03-13 00:00:00.000 0
2628192      373133.309400      1995-02-22 00:00:00.000 0
993600      371407.459500      1995-03-05 00:00:00.000 0
2300070      367371.145200      1995-03-13 00:00:00.000 0

```

(10 row(s) affected)

**QUERY 4 FOR VALIDATION**

```

-----
-- using default substitutions

/* TPC_H Query 4 - Order Priority Checking */

```

```

SELECT O_ORDERPRIORITY,
COUNT(*) AS ORDER_COUNT
FROM ORDERS
WHERE O_ORDERDATE >= '1993-07-01' AND
O_ORDERDATE < dateadd (mm, 3, '1993-07-01') AND
EXISTS (SELECT *
FROM LINEITEM
WHERE L_ORDERKEY = O_ORDERKEY AND
L_COMMITDATE < L_RECEIPTDATE
)
GROUP BY O_ORDERPRIORITY
ORDER BY O_ORDERPRIORITY

```

O_ORDERPRIORITY	ORDER_COUNT
1-URGENT	10594
2-HIGH	10476
3-MEDIUM	10410
4-NOT SPECIFIED	10556
5-LOW	10487

(5 row(s) affected)

**QUERY 5 FOR VALIDATION**

```

-----
-- using default substitutions

/* TPC_H Query 5 - Local Supplier Volume */

```

```

SELECT N_NAME,
SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) AS REVENUE

```

```

FROMCUSTOMER,
ORDERS,
LINEITEM,
SUPPLIER,
NATION,
REGION
WHEREC_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 = 'ASIA' AND
O_ORDERDATE >= '1994-01-01' AND
O_ORDERDATE < DATEADD(Y, 1, '1994-01-01')
GROUPBYN_NAME
ORDERBYREVENUE DESC

```

N_NAME	REVENUE
INDONESIA	55502041.169700
VIETNAM	55295086.996700
CHINA	53724494.256600
INDIA	52035512.000200
JAPAN	45410175.695400

(5 row(s) affected)

**QUERY 6 FOR VALIDATION**

-- using default substitutions

/\* TPC\_H Query 6 - Forecasting Revenue Change \*/

```

SELECTSUM(L_EXTENDEDPRICE*L_DISCOUNT) AS REVENUE
FROMLINEITEM
WHEREL_SHIPDATE >= '1994-01-01' AND
L_SHIPDATE < dateadd (yy, 1, '1994-01-01') AND
L_DISCOUNT BETWEEN .06 - 0.01 AND .06 + 0.01 AND
L_QUANTITY < 24

```

```

REVENUE
-----
123141078.228300

```

(1 row(s) affected)

**QUERY 7 FOR VALIDATION**

-- using default substitutions

/\* TPC\_H Query 7 - Volume Shipping \*/

```

SELECTSUPP_NATION,
CUST_NATION,
L_YEAR,
SUM(VOLUME) AS REVENUE
FROM(SELECTN1.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
FROMSUPPLIER,
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 = 'GERMANY')
OR
(N1.N_NAME = 'GERMANY' AND N2.N_NAME = 'FRANCE'))
) AND
L_SHIPDATE BETWEEN '1995-01-01' AND '1996-12-31'
)AS SHIPPING
GROUPBYSUPP_NATION,
CUST_NATION,
L_YEAR
ORDERBYSUPP_NATION,
CUST_NATION,
L_YEAR

```

SUPP_NATION	CUST_NATION	L_YEAR	REVENUE
FRANCE	GERMANY	1995	54639732.733600
FRANCE	GERMANY	1996	54633083.307600
GERMANY	FRANCE	1995	52531746.669700
GERMANY	FRANCE	1996	52520549.022400

(4 row(s) affected)

**QUERY 8 FOR VALIDATION**

-- using default substitutions

/\* TPC\_H Query 8 - National Market Share \*/

```

SELECTO_YEAR,
SUM(CASE WHENNATION = 'BRAZIL'
THENVOLUME
ELSE0
END) / SUM(VOLUME) AS MKT_SHARE

```

```

FROM (SELECT datepart (yy, O_ORDERDATE) AS O_YEAR,
L_EXTENDEDPRI * (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 = 'ECONOMY ANODIZED STEEL'
) AS ALL_NATIONS
GROUP BY O_YEAR
ORDER BY O_YEAR

```

```

O_YEAR      MKT_SHARE
-----
1995        0.034436
1996        0.041486

```

(2 row(s) affected)

#### QUERY 9 FOR VALIDATION

```
-- using default substitutions
```

```
/* 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_EXTENDEDPRI * (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 '%green%'
) AS PROFIT
GROUP BY NATION,
O_YEAR
ORDER BY NATION,
O_YEAR DESC

```

NATION	O_YEAR	SUM_PROFIT
ALGERIA	1998	31342867.234500
ALGERIA	1997	57138193.023300
ALGERIA	1996	56140140.133000
ALGERIA	1995	53051469.653400
ALGERIA	1994	53867582.128600
ALGERIA	1993	54942718.132400
ALGERIA	1992	54628034.712700
ARGENTINA	1998	30211185.708100
ARGENTINA	1997	50805741.752300
ARGENTINA	1996	51923746.575500
ARGENTINA	1995	49298625.766600
ARGENTINA	1994	50835610.109500
ARGENTINA	1993	51646079.177500
ARGENTINA	1992	50410314.994800
BRAZIL	1998	27217924.383200
BRAZIL	1997	48378669.198900
BRAZIL	1996	50482870.357200
BRAZIL	1995	47623383.634900
BRAZIL	1994	47840165.725600
BRAZIL	1993	49054694.035100
BRAZIL	1992	48667639.084200
CANADA	1998	30379833.768500
CANADA	1997	50465052.311400
CANADA	1996	52560501.390400
CANADA	1995	52375332.809200
CANADA	1994	52600364.658700
CANADA	1993	52644504.073500
CANADA	1992	53932871.697000
CHINA	1998	31075466.164900
CHINA	1997	50551874.449900
CHINA	1996	51039293.875400
CHINA	1995	49287534.616900
CHINA	1994	50851090.067400
CHINA	1993	54229629.833000
CHINA	1992	52400529.372000
EGYPT	1998	29054433.385600
EGYPT	1997	50627611.452400
EGYPT	1996	49542212.844600
EGYPT	1995	48311550.320700
EGYPT	1994	49790644.736000
EGYPT	1993	48904292.969200

EGYPT	1992	49434932.619200	JAPAN	1995	43749356.399600
ETHIOPIA	1998	28040717.267400	JAPAN	1994	44840243.070000
ETHIOPIA	1997	47455009.866400	JAPAN	1993	44660015.532800
ETHIOPIA	1996	46491097.572900	JAPAN	1992	45410249.121600
ETHIOPIA	1995	46804449.300900	JORDAN	1998	26901488.578100
ETHIOPIA	1994	48516143.917000	JORDAN	1997	45471878.409900
ETHIOPIA	1993	46551891.562900	JORDAN	1996	46794325.791800
ETHIOPIA	1992	44934648.642800	JORDAN	1995	45178828.576000
FRANCE	1998	32226407.839200	JORDAN	1994	45333636.507600
FRANCE	1997	47121485.860200	JORDAN	1993	47971496.097900
FRANCE	1996	47263135.496000	JORDAN	1992	44717239.177400
FRANCE	1995	47275997.570700	KENYA	1998	28597614.337000
FRANCE	1994	47067209.331500	KENYA	1997	47949733.727400
FRANCE	1993	51163370.106200	KENYA	1996	46886924.622900
FRANCE	1992	47846235.331300	KENYA	1995	46072338.755200
GERMANY	1998	28624942.659800	KENYA	1994	45772061.171100
GERMANY	1997	49309074.877300	KENYA	1993	46308728.234500
GERMANY	1996	49918683.168400	KENYA	1992	47257780.840600
GERMANY	1995	52650718.724200	MOROCCO	1998	26732115.580000
GERMANY	1994	50346900.422600	MOROCCO	1997	45637304.249900
GERMANY	1993	50991895.805900	MOROCCO	1996	45558221.745100
GERMANY	1992	48274126.099100	MOROCCO	1995	47851318.887400
INDIA	1998	29943144.354400	MOROCCO	1994	46272172.944500
INDIA	1997	50665453.231200	MOROCCO	1993	46764326.181800
INDIA	1996	50283092.291200	MOROCCO	1992	48122783.583100
INDIA	1995	50006774.644600	MOZAMBIQUE	1998	30712392.011200
INDIA	1994	48995190.755600	MOZAMBIQUE	1997	50316528.762500
INDIA	1993	50286902.852500	MOZAMBIQUE	1996	51640320.250900
INDIA	1992	50850329.402300	MOZAMBIQUE	1995	50693774.506000
INDONESIA	1998	27672339.996500	MOZAMBIQUE	1994	49253277.626000
INDONESIA	1997	50512145.725600	MOZAMBIQUE	1993	49153016.536700
INDONESIA	1996	51653060.116600	MOZAMBIQUE	1992	48247551.850300
INDONESIA	1995	51508779.594000	PERU	1998	29326102.319600
INDONESIA	1994	52817950.321800	PERU	1997	49753780.395100
INDONESIA	1993	47959994.955100	PERU	1996	50935170.292700
INDONESIA	1992	51776605.032300	PERU	1995	53309883.407200
IRAN	1998	29065736.238100	PERU	1994	50643531.796800
IRAN	1997	50042063.054500	PERU	1993	51584622.002000
IRAN	1996	50926653.188100	PERU	1992	47523899.054700
IRAN	1995	51249667.648500	ROMANIA	1998	30368667.399900
IRAN	1994	50337085.865000	ROMANIA	1997	50365683.852600
IRAN	1993	51730763.490200	ROMANIA	1996	49598999.014800
IRAN	1992	49955856.563400	ROMANIA	1995	47537642.869700
IRAQ	1998	31624551.001700	ROMANIA	1994	51455283.009500
IRAQ	1997	55121749.019500	ROMANIA	1993	50407136.892200
IRAQ	1996	55897663.793700	ROMANIA	1992	48185385.128600
IRAQ	1995	54815472.517000	RUSSIA	1998	28322384.026600
IRAQ	1994	54408516.126600	RUSSIA	1997	50106685.182200
IRAQ	1993	53633167.977000	RUSSIA	1996	51753342.430200
IRAQ	1992	55891939.339600	RUSSIA	1995	49215820.364700
JAPAN	1998	27934179.669500	RUSSIA	1994	52205666.440700
JAPAN	1997	44517162.546300	RUSSIA	1993	51860230.034000
JAPAN	1996	42545606.120400	RUSSIA	1992	53251677.153000



SAUDI ARABIA	1998	31541259.810000
SAUDI ARABIA	1997	52438750.807600
SAUDI ARABIA	1996	52543737.819700
SAUDI ARABIA	1995	52938696.533100
SAUDI ARABIA	1994	51389601.966800
SAUDI ARABIA	1993	52937508.881800
SAUDI ARABIA	1992	54843459.641400
UNITED KINGDOM	1998	28494874.004000
UNITED KINGDOM	1997	49381810.898600
UNITED KINGDOM	1996	51386853.960400
UNITED KINGDOM	1995	51509586.788500
UNITED KINGDOM	1994	48086499.711500
UNITED KINGDOM	1993	49166827.223500
UNITED KINGDOM	1992	49349122.082500
UNITED STATES	1998	25126238.946100
UNITED STATES	1997	50077306.418600
UNITED STATES	1996	48048649.470300
UNITED STATES	1995	48809032.422600
UNITED STATES	1994	49296747.182700
UNITED STATES	1993	48029946.801400
UNITED STATES	1992	48671944.498300
VIETNAM	1998	30442736.059400
VIETNAM	1997	50309179.794200
VIETNAM	1996	50488161.410000
VIETNAM	1995	49658284.612500
VIETNAM	1994	50596057.260700
VIETNAM	1993	50953919.151900
VIETNAM	1992	49613838.315100

(175 row(s) affected)

QUERY 10 FOR VALIDATION

-- using default substitutions

/\* TPC\_H Query 10 - Returned Item Reporting \*/

```

SELECT TOP 20
  C_CUSTKEY,
  C_NAME,
  SUM(L_EXTENDEDPRI*(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 >= '1993-10-01' AND

```

```

O_ORDERDATE < dateadd(mm, 3, '1993-10-01') 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

```

C_CUSTKEY	C_NAME	REVENUE	C_ACCTBAL
57040	Customer#000057040	734235.245500	632.870000
JAPAN	Eioyzjf4pp		22-895-641-
3466	requests sleep blithely about the furiously i		
143347	Customer#000143347	721002.694800	2557.470000
EGYPT	1aReFYv,Kw4		14-742-935-
3718	fluffily bold excuses haggle finally after the u		
60838	Customer#000060838	679127.307700	2454.770000
BRAZIL	64EaJ5vMAHWJlBOxJklpNc2RjiWE		12-913-494-
9813	furiously even pinto beans integrate under the ruthless foxes; ironic,		
even dolphins across the slyl			
101998	Customer#000101998	637029.566700	3790.890000
UNITED KINGDOM	01c9CILnNtfOQYmZj		33-593-865-
6378	accounts doze blithely! enticing, final deposits sleep blithely special		
accounts. slyly express accounts pla			
125341	Customer#000125341	633508.086000	4983.510000
GERMANY	S29ODD6bceU8QSuEJznkNaK		17-582-695-
5962	quickly express requests wake quickly blithely		
25501	Customer#000025501	620269.784900	7725.040000
ETHIOPIA	W556MXuoiaYCCZamJI,Rn0B4ACUGdkQ8DZ		15-874-808-
6793	quickly special requests sleep evenly among the special deposits. special		
deposi			
115831	Customer#000115831	596423.867200	5098.100000
FRANCE	rFeBbEEyk dl ne7zV5fDrmiqloK09wV7pxqCgIc		16-715-386-
3788	carefully bold excuses sleep alongside of the thinly idle		
84223	Customer#000084223	594998.023900	528.650000
UNITED KINGDOM	nAVZCs6BaWap rrM27N 2qBnzc5WBauxba		33-442-824-
8191	pending, final ideas haggle final requests. unusual, regular asymptotes		
affix according to the even foxes.			
54289	Customer#000054289	585603.391800	5583.020000
IRAN	vXCxoCsU0Bad5JQI ,oobkZ		20-834-292-
4707	express requests sublata blithely regular requests. regular, even ideas		
solve.			
39922	Customer#000039922	584878.113400	7321.110000
GERMANY	Zgy4s5012GKN4pLDPBU8m342gIw6R		17-147-757-

```

8036 even pinto beans haggle. slyly bold accounts inte
6226 Customer#000006226 576783.760600 2230.090000
UNITED KINGDOM 8gPu8,NPGkfyQQ0hcIYUGPIBWC,ybP5g, 33-657-701-
3391 quickly final requests against the regular instructions wake blithely
final instructions. pa
922 Customer#000000922 576767.533300 3869.250000
GERMANY Az9RFaut7NkPnc5zSD2PwHgVwr4jRzq 17-945-916-
9648 boldly final requests cajole blith
147946 Customer#000147946 576455.132000 2030.130000
ALGERIA iANyZHjqhyy7AjahOpTrYyhJ 10-886-956-
3143 furiously even accounts are blithely above the furiousl
115640 Customer#000115640 569341.193300 6436.100000
ARGENTINA Vtgfia9qI 7EpHgecU1X 11-411-543-
4901 final instructions are slyly according to the
73606 Customer#000073606 568656.857800 1785.670000
JAPAN xuR0Tro5yChDfOCrjkd2ol 22-437-653-
6966 furiously bold orbits about the furiously busy requests wake across the
furiously quiet theodolites. d
110246 Customer#000110246 566842.981500 7763.350000
VIETNAM 7KzflgX MDOq7sOkI 31-943-426-
9837 dolphins sleep blithely among the slyly final
142549 Customer#000142549 563537.236800 5085.990000
INDONESIA ChqEoK43OysjdHbtKCp6dKqjNyvvi9 19-955-562-
2398 regular, unusual dependencies boost slyly; ironic attainments nag fluffily
into the unusual packages?
146149 Customer#000146149 557254.986500 1791.550000
ROMANIA s87fvzFQpU 29-744-164-
6487 silent, unusual requests detect quickly slyly regul
52528 Customer#000052528 556397.350900 551.790000
ARGENTINA NFztyTOR10UOJ 11-208-192-
3205 unusual requests detect. slyly dogged theodolites use slyly. deposit
23431 Customer#000023431 554269.536000 3381.860000
ROMANIA HgiV0phqhaIa9aydNoIlb 29-915-458-
2654 instructions nag quickly. furiously bold accounts cajol

```

```

(SELECT SUM(PS_SUPPLYCOST*PS_AVAILQTY) * 0.0001000000
FROM PARTSUPP,
SUPPLIER,
NATION
WHERE PS_SUPPKEY = S_SUPPKEY AND
S_NATIONKEY = N_NATIONKEY AND
N_NAME = 'GERMANY'
)
ORDER BY VALUE DESC

```

PS_PARTKEY	VALUE
129760	17538456.860000
166726	16503353.920000
191287	16474801.970000
161758	16101755.540000
34452	15983844.720000
139035	15907078.340000
9403	15451755.620000
154358	15212937.880000
38823	15064802.860000
85606	15053957.150000
33354	14408297.400000
154747	14407580.680000
82865	14235489.780000
76094	14094247.040000
222	13937777.740000
121271	13908336.000000
55221	13716120.470000
22819	13666434.280000
76281	13646853.680000
85298	13581154.930000
85158	13554904.000000
139684	13535538.720000
31034	13498025.250000
87305	13482847.040000
10181	13445148.750000
62323	13411824.300000
26489	13377256.380000
96493	13339057.830000
56548	13329014.970000
55576	13306843.350000
159751	13306614.480000
92406	13287414.500000
182636	13223726.740000
199969	13135288.210000
62865	13001926.940000
7284	12945298.190000
197867	12944510.520000
11562	12931575.510000
75165	12916918.120000
97175	12911283.500000
140840	12896562.230000

(20 row(s) affected)

**QUERY 11 FOR VALIDATION**

-- using default substitutions

/\* TPC\_H Query 11 - Important Stock Identification \*/

```

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 = 'GERMANY'
GROUP BY PS_PARTKEY
HAVING SUM(PS_SUPPLYCOST*PS_AVAILQTY) >

```

65241	12890600.460000
166120	12876927.220000
9035	12863828.700000
144616	12853549.300000
176723	12832309.740000
170884	12792136.580000
29790	12723300.330000
95213	12555483.730000
183873	12550533.050000
171235	12476538.300000
21533	12437821.320000
17290	12432159.500000
156397	12260623.500000
122611	12222812.980000
139155	12220319.250000
146316	12215800.610000
171381	12199734.520000
198633	12078226.950000
167417	12046637.620000
59512	12043468.760000
31688	12034893.640000
159586	12001505.840000
8993	11963814.300000
120302	11857707.550000
43536	11779340.520000
9552	11776909.160000
86223	11772205.080000
53776	11758669.650000
131285	11616953.740000
91628	11611114.830000
169644	11567959.720000
182299	11567462.050000
33107	11453818.760000
104184	11436657.440000
67027	11419127.140000
176869	11371451.710000
30885	11369674.790000
54420	11345076.880000
72240	11313951.050000
178708	11294635.170000
81298	11273686.130000
158324	11243442.720000
117095	11242535.240000
176793	11237733.380000
86091	11177793.790000
116033	11145434.360000
129058	11119112.200000
193714	11104706.390000
117195	11077217.960000
49851	11043701.780000
19791	11030662.620000
75800	11012401.620000
161562	10996371.690000

10119	10980015.750000
39185	10970042.560000
47223	10950022.130000
175594	10942923.050000
111295	10893675.610000
155446	10852764.570000

+++++ Part of query output removed +++++

92516	8003836.800000
30802	8003710.880000
107418	8000430.300000
46620	7999778.350000
191803	7994734.150000
106343	7993087.760000
59362	7990397.460000
8329	7990052.900000
75133	7988244.000000
179023	7986829.620000
135899	7985726.640000
5824	7985340.020000
148579	7984889.560000
95888	7984735.720000
9791	7982699.790000
170437	7982370.720000
39782	7977858.240000
20605	7977556.000000
28682	7976960.000000
42172	7973399.000000
56137	7971405.400000
64729	7970769.720000
98643	7968603.730000
153787	7967535.580000
8932	7967222.190000
20134	7965713.280000
197635	7963507.580000
80408	7963312.170000
37728	7961875.680000
26624	7961772.310000
44736	7961144.100000
29763	7960605.030000
36147	7959463.680000
146040	7957587.660000
115469	7957485.140000
142276	7956790.630000
181280	7954037.350000
115096	7953047.550000
109650	7952258.730000
93862	7951992.240000
158325	7950728.300000
55952	7950387.060000

```

122397      7947106.270000
28114       7946945.720000
11966       7945197.480000
47814       7944083.000000
85096       7943691.060000
51657       7943593.770000
196680      7943578.890000
13141       7942730.340000
193327      7941036.250000
152612      7940663.710000
139680      7939242.360000
31134       7938318.300000
45636       7937240.850000
56694       7936015.950000
8114        7933921.880000
71518       7932261.690000
72922       7930400.640000
146699      7929167.400000
92387       7928972.670000
186289      7928786.190000
95952       7927972.780000
196514      7927180.700000
4403        7925729.040000
2267        7925649.370000
45924       7925047.680000
11493       7916722.230000
104478      7916253.600000
166794      7913842.000000
161995      7910874.270000
23538       7909752.060000
41093       7909579.920000
112073      7908617.570000
92814       7908262.500000
88919       7907992.500000
79753       7907933.880000
108765      7905338.980000
146530      7905336.600000
71475       7903367.580000
36289       7901946.500000
61739       7900794.000000
52338       7898638.080000
194299      7898421.240000
105235      7897829.940000
77207       7897752.720000
96712       7897575.270000
10157       7897046.250000
171154      7896814.500000
79373       7896186.000000
113808      7893353.880000
27901       7892952.000000
128820      7892882.720000
25891       7890511.200000
122819      7888881.020000

```

```

154731      7888301.330000
101674      7879324.600000
51968       7879102.210000
72073       7877736.110000
5182        7874521.730000

```

(1048 row(s) affected)

**QUERY 12 FOR VALIDATION**

-----  
-- using default substitutions

/\* TPC\_H Query 12 - Shipping Modes and Order Priority \*/

```

SELECT L_SHIPMODE,
SUM(CASEWHEN O_ORDERPRIORITY = '1-URGENT' OR
O_ORDERPRIORITY = '2-HIGH'
THEN 1
ELSE 0
END) AS HIGH_LINE_COUNT,
SUM(CASEWHEN 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 ('MAIL','SHIP') 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

```

```

L_SHIPMODE HIGH_LINE_COUNT LOW_LINE_COUNT
-----
MAIL          6202             9324
SHIP          6200             9262

```

(2 row(s) affected)

**QUERY 13 FOR VALIDATION**

-----  
-- using default substitutions

/\* TPC\_H Query 13 - Customer Distribution \*/

```

SELECT C_COUNT,
COUNT(*) AS CUSTDIST
FROM (SELECT C_CUSTKEY,
COUNT(O_ORDERKEY)

```

```

FROMCUSTOMER left outer join ORDERS on
  C_CUSTKEY = O_CUSTKEY AND
  O_COMMENT not like '%special%requests%'
GROUPBYC_CUSTKEY
)AS C_ORDERS (C_CUSTKEY, C_COUNT)
GROUPBYC_COUNT
ORDERBYCUSTDISTDESC,
  C_COUNT DESC

```

C_COUNT	CUSTDIST
0	50004
9	6641
10	6566
11	6058
8	5949
12	5553
13	4989
19	4748
7	4707
18	4625
15	4552
17	4530
14	4484
20	4461
16	4323
21	4217
22	3730
6	3334
23	3129
24	2622
25	2079
5	1972
26	1593
27	1185
4	1033
28	869
29	559
3	398
30	373
31	235
2	144
32	128
33	71
34	48
35	33
1	23
36	17
37	7
40	4
38	4
39	2
41	1

(42 row(s) affected)

**QUERY 14 FOR VALIDATION**

-----  
-- using default substitutions

/\* TPC\_H Query 14 - Promotion Effect \*/

```

SELECT100.00 * SUM (CASEWHEN P_TYPE LIKE 'PROMO%'
  THEN L_EXTENDEDPRI* (1-L_DISCOUNT)
  ELSE 0
END) / SUM(L_EXTENDEDPRI*(1-L_DISCOUNT)) AS PROMO_REVENUE
FROMLINEITEM,
  PART
WHEREL_PARTKEY = P_PARTKEY AND
  L_SHIPDATE >= '1995-09-01' AND
  L_SHIPDATE < dateadd(mm, 1, '1995-09-01')

```

PROMO\_REVENUE

-----  
16.380779

(1 row(s) affected)

**QUERY 15 FOR VALIDATION**

-----  
-- using default substitutions

/\* TPC\_H Query 15 - Create View for Top Supplier Query \*/

```

CREATEVIEW REVENUE0 (SUPPLIER_NO, TOTAL_REVENUE)
AS
SELECTL_SUPPKEY,
  SUM(L_EXTENDEDPRI*(1-L_DISCOUNT))
FROMLINEITEM
WHEREL_SHIPDATE >= '1996-01-01' AND
  L_SHIPDATE < dateadd(mm, 3, '1996-01-01')
GROUPBYL_SUPPKEY
GO

```

/\* TPC\_H Query 15 - Top Supplier \*/

```

SELECTS_SUPPKEY,
  S_NAME,
  S_ADDRESS,
  S_PHONE,
  TOTAL_REVENUE
FROMSUPPLIER,
  REVENUE0
WHERE S_SUPPKEY = SUPPLIER_NO AND
  TOTAL_REVENUE = (SELECTMAX(TOTAL_REVENUE))

```

```

FROMREVENUE0
)
ORDERBYS_SUPPKEY
DROP VIEW REVENUE0

S_SUPPKEY  S_NAME          S_ADDRESS
S_PHONE    TOTAL_REVENUE
-----
8449      Supplier#000008449      Wp34zim9qYFbVctdW
20-469-856-8873 1772627.208700

```

(1 row(s) affected)

QUERY 16 FOR VALIDATION

-- using default substitutions

/\* 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#45' AND
P_TYPE NOT LIKE 'MEDIUM POLISHED%' AND
P_SIZE IN (49, 14, 23, 45, 19, 3, 36, 9) 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

```

P_BRAND	P_TYPE	P_SIZE	SUPPLIER_CNT
Brand#41	MEDIUM BRUSHED TIN	3	28
Brand#54	STANDARD BRUSHED COPPER	14	27
Brand#11	STANDARD BRUSHED TIN	23	24
Brand#11	STANDARD BURNISHED BRASS	36	24
Brand#15	MEDIUM ANODIZED NICKEL	3	24
Brand#15	SMALL ANODIZED BRASS	45	24
Brand#15	SMALL BURNISHED NICKEL	19	24
Brand#21	MEDIUM ANODIZED COPPER	3	24

Brand#22	SMALL BRUSHED NICKEL	3	24
Brand#22	SMALL BURNISHED BRASS	19	24
Brand#25	MEDIUM BURNISHED COPPER	36	24
Brand#31	PROMO POLISHED COPPER	36	24
Brand#33	LARGE POLISHED TIN	23	24
Brand#33	PROMO POLISHED STEEL	14	24
Brand#35	PROMO BRUSHED NICKEL	14	24
Brand#41	ECONOMY BRUSHED STEEL	9	24
Brand#41	ECONOMY POLISHED TIN	19	24
Brand#41	LARGE PLATED COPPER	36	24
Brand#42	ECONOMY PLATED BRASS	3	24
Brand#42	STANDARD POLISHED TIN	49	24
Brand#43	PROMO BRUSHED TIN	3	24
Brand#43	SMALL ANODIZED COPPER	36	24
Brand#44	STANDARD POLISHED NICKEL	3	24
Brand#52	ECONOMY PLATED TIN	14	24
Brand#52	STANDARD BURNISHED NICKEL	3	24
Brand#53	MEDIUM ANODIZED STEEL	14	24
Brand#14	PROMO ANODIZED NICKEL	45	23
Brand#32	ECONOMY PLATED BRASS	9	23
Brand#52	SMALL ANODIZED COPPER	3	23
Brand#11	ECONOMY BRUSHED COPPER	45	20
Brand#11	ECONOMY PLATED BRASS	23	20
Brand#11	LARGE BRUSHED COPPER	49	20
Brand#11	LARGE POLISHED COPPER	49	20
Brand#12	STANDARD ANODIZED TIN	49	20
Brand#12	STANDARD PLATED BRASS	19	20
Brand#13	ECONOMY BRUSHED BRASS	9	20
Brand#13	ECONOMY BURNISHED STEEL	14	20
Brand#13	LARGE BURNISHED NICKEL	19	20
Brand#13	MEDIUM BURNISHED COPPER	36	20
Brand#13	SMALL BRUSHED TIN	45	20
Brand#13	STANDARD ANODIZED COPPER	3	20
Brand#13	STANDARD PLATED NICKEL	23	20
Brand#14	ECONOMY ANODIZED COPPER	14	20
Brand#14	ECONOMY PLATED TIN	36	20
Brand#14	ECONOMY POLISHED NICKEL	3	20
Brand#14	MEDIUM ANODIZED NICKEL	3	20
Brand#14	SMALL POLISHED TIN	14	20
Brand#15	MEDIUM ANODIZED COPPER	9	20
Brand#15	MEDIUM PLATED TIN	23	20
Brand#15	PROMO PLATED BRASS	14	20
Brand#15	SMALL ANODIZED COPPER	45	20
Brand#15	SMALL PLATED COPPER	49	20
Brand#15	STANDARD PLATED TIN	3	20
Brand#21	LARGE ANODIZED COPPER	36	20
Brand#21	LARGE BRUSHED TIN	3	20
Brand#21	MEDIUM ANODIZED COPPER	14	20
Brand#21	PROMO BRUSHED TIN	36	20
Brand#21	PROMO POLISHED NICKEL	45	20
Brand#21	SMALL ANODIZED COPPER	9	20
Brand#21	SMALL POLISHED NICKEL	23	20
Brand#22	LARGE ANODIZED COPPER	36	20

Brand#22	LARGE BRUSHED COPPER	49	20
Brand#22	PROMO ANODIZED TIN	49	20
Brand#22	PROMO POLISHED BRASS	45	20
Brand#22	SMALL BURNISHED STEEL	45	20
Brand#23	MEDIUM ANODIZED STEEL	45	20
Brand#23	PROMO POLISHED STEEL	23	20
Brand#23	STANDARD BRUSHED TIN	14	20
Brand#23	STANDARD PLATED NICKEL	36	20
Brand#24	PROMO PLATED COPPER	49	20
Brand#24	PROMO PLATED STEEL	49	20
Brand#24	PROMO POLISHED STEEL	9	20
Brand#24	STANDARD BRUSHED TIN	36	20
Brand#25	LARGE ANODIZED BRASS	3	20
Brand#25	PROMO BURNISHED TIN	3	20
Brand#31	ECONOMY POLISHED NICKEL	3	20
Brand#31	MEDIUM PLATED TIN	45	20
Brand#31	SMALL ANODIZED STEEL	14	20
Brand#32	ECONOMY ANODIZED COPPER	36	20
Brand#32	ECONOMY BRUSHED NICKEL	49	20
Brand#32	LARGE ANODIZED TIN	19	20
Brand#32	MEDIUM BURNISHED COPPER	19	20
Brand#32	SMALL ANODIZED STEEL	45	20
Brand#33	ECONOMY POLISHED COPPER	19	20
Brand#33	PROMO PLATED NICKEL	14	20
Brand#33	SMALL POLISHED TIN	9	20
Brand#33	STANDARD ANODIZED BRASS	49	20
Brand#33	STANDARD BURNISHED BRASS	45	20
Brand#34	ECONOMY BRUSHED NICKEL	49	20
Brand#34	LARGE BRUSHED BRASS	19	20
Brand#34	SMALL BRUSHED TIN	3	20
Brand#34	STANDARD PLATED COPPER	9	20
Brand#35	LARGE ANODIZED NICKEL	3	20
Brand#35	MEDIUM ANODIZED BRASS	45	20
Brand#35	MEDIUM ANODIZED STEEL	23	20
Brand#35	PROMO ANODIZED COPPER	49	20
Brand#35	SMALL POLISHED COPPER	14	20
Brand#41	LARGE ANODIZED STEEL	3	20
Brand#41	LARGE BRUSHED NICKEL	23	20
Brand#41	LARGE BURNISHED COPPER	3	20

+++++ Part of query output removed +++++

Brand#55	SMALL POLISHED COPPER	45	4
Brand#55	SMALL POLISHED COPPER	49	4
Brand#55	SMALL POLISHED NICKEL	9	4
Brand#55	SMALL POLISHED NICKEL	14	4
Brand#55	SMALL POLISHED NICKEL	19	4
Brand#55	SMALL POLISHED NICKEL	23	4
Brand#55	SMALL POLISHED NICKEL	45	4
Brand#55	SMALL POLISHED NICKEL	49	4
Brand#55	SMALL POLISHED STEEL	19	4

Brand#55	SMALL POLISHED STEEL	45	4
Brand#55	SMALL POLISHED TIN	14	4
Brand#55	SMALL POLISHED TIN	23	4
Brand#55	SMALL POLISHED TIN	45	4
Brand#55	STANDARD ANODIZED BRASS	9	4
Brand#55	STANDARD ANODIZED BRASS	23	4
Brand#55	STANDARD ANODIZED BRASS	49	4
Brand#55	STANDARD ANODIZED COPPER	9	4
Brand#55	STANDARD ANODIZED COPPER	14	4
Brand#55	STANDARD ANODIZED COPPER	45	4
Brand#55	STANDARD ANODIZED NICKEL	3	4
Brand#55	STANDARD ANODIZED NICKEL	14	4
Brand#55	STANDARD ANODIZED NICKEL	45	4
Brand#55	STANDARD ANODIZED NICKEL	49	4
Brand#55	STANDARD ANODIZED STEEL	3	4
Brand#55	STANDARD ANODIZED STEEL	14	4
Brand#55	STANDARD ANODIZED TIN	14	4
Brand#55	STANDARD ANODIZED TIN	36	4
Brand#55	STANDARD ANODIZED TIN	45	4
Brand#55	STANDARD BRUSHED BRASS	9	4
Brand#55	STANDARD BRUSHED BRASS	19	4
Brand#55	STANDARD BRUSHED COPPER	14	4
Brand#55	STANDARD BRUSHED COPPER	19	4
Brand#55	STANDARD BRUSHED NICKEL	3	4
Brand#55	STANDARD BRUSHED NICKEL	36	4
Brand#55	STANDARD BRUSHED STEEL	9	4
Brand#55	STANDARD BRUSHED STEEL	14	4
Brand#55	STANDARD BRUSHED STEEL	19	4
Brand#55	STANDARD BRUSHED STEEL	49	4
Brand#55	STANDARD BRUSHED TIN	19	4
Brand#55	STANDARD BRUSHED TIN	49	4
Brand#55	STANDARD BURNISHED BRASS	9	4
Brand#55	STANDARD BURNISHED BRASS	19	4
Brand#55	STANDARD BURNISHED BRASS	23	4
Brand#55	STANDARD BURNISHED BRASS	36	4
Brand#55	STANDARD BURNISHED COPPER	3	4
Brand#55	STANDARD BURNISHED NICKEL	9	4
Brand#55	STANDARD BURNISHED NICKEL	49	4
Brand#55	STANDARD BURNISHED STEEL	19	4
Brand#55	STANDARD BURNISHED STEEL	23	4
Brand#55	STANDARD BURNISHED STEEL	36	4
Brand#55	STANDARD BURNISHED STEEL	45	4
Brand#55	STANDARD BURNISHED TIN	9	4
Brand#55	STANDARD BURNISHED TIN	19	4
Brand#55	STANDARD BURNISHED TIN	36	4
Brand#55	STANDARD BURNISHED TIN	49	4
Brand#55	STANDARD PLATED BRASS	9	4
Brand#55	STANDARD PLATED BRASS	45	4
Brand#55	STANDARD PLATED BRASS	49	4
Brand#55	STANDARD PLATED COPPER	9	4
Brand#55	STANDARD PLATED COPPER	45	4
Brand#55	STANDARD PLATED NICKEL	3	4
Brand#55	STANDARD PLATED NICKEL	19	4

```

Brand#55 STANDARD PLATED NICKEL 45 4
Brand#55 STANDARD PLATED STEEL 14 4
Brand#55 STANDARD PLATED STEEL 23 4
Brand#55 STANDARD PLATED STEEL 49 4
Brand#55 STANDARD PLATED TIN 9 4
Brand#55 STANDARD PLATED TIN 14 4
Brand#55 STANDARD PLATED TIN 36 4
Brand#55 STANDARD POLISHED BRASS 3 4
Brand#55 STANDARD POLISHED BRASS 9 4
Brand#55 STANDARD POLISHED BRASS 23 4
Brand#55 STANDARD POLISHED COPPER 3 4
Brand#55 STANDARD POLISHED COPPER 23 4
Brand#55 STANDARD POLISHED COPPER 45 4
Brand#55 STANDARD POLISHED NICKEL 3 4
Brand#55 STANDARD POLISHED NICKEL 23 4
Brand#55 STANDARD POLISHED NICKEL 36 4
Brand#55 STANDARD POLISHED NICKEL 45 4
Brand#55 STANDARD POLISHED NICKEL 49 4
Brand#55 STANDARD POLISHED STEEL 14 4
Brand#55 STANDARD POLISHED STEEL 23 4
Brand#55 STANDARD POLISHED TIN 9 4
Brand#55 STANDARD POLISHED TIN 19 4
Brand#55 STANDARD POLISHED TIN 36 4
Brand#11 SMALL BRUSHED TIN 19 3
Brand#15 LARGE PLATED NICKEL 45 3
Brand#15 LARGE POLISHED NICKEL 9 3
Brand#21 PROMO BURNISHED STEEL 45 3
Brand#22 STANDARD PLATED STEEL 23 3
Brand#25 LARGE PLATED STEEL 19 3
Brand#32 STANDARD ANODIZED COPPER 23 3
Brand#33 SMALL ANODIZED BRASS 9 3
Brand#35 MEDIUM ANODIZED TIN 19 3
Brand#51 SMALL PLATED BRASS 23 3
Brand#52 MEDIUM BRUSHED BRASS 45 3
Brand#53 MEDIUM BRUSHED TIN 45 3
Brand#54 ECONOMY POLISHED BRASS 9 3
Brand#55 PROMO PLATED BRASS 19 3
Brand#55 STANDARD PLATED TIN 49 3

```

(18314 row(s) affected)

**QUERY 17 FOR VALIDATION**

-- using default substitutions

/\* TPC\_H Query 17 - Small-Quantity-Order Revenue \*/

```

SELECT SUM(L_EXTENDEDPRI) / 7.0 AS AVG_YEARLY
FROM LINEITEM,
PART
WHERE P_PARTKEY = L_PARTKEY AND
P_BRAND = 'Brand#23' AND
P_CONTAINER = 'MED BOX' AND

```

```

L_QUANTITY < (SELECT 0.2 * AVG(L_QUANTITY)
FROM LINEITEM
WHERE L_PARTKEY = P_PARTKEY
)

```

AVG\_YEARLY

-----  
348406.054286

(1 row(s) affected)

**QUERY 18 FOR VALIDATION**

-- using default substitutions

/\* 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) > 300
) 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

```

C_NAME	C_CUSTKEY	O_ORDERKEY	O_ORDERDATE
Customer#000128120	128120	4722021	1994-04-07
00:00:00.000 544089.090000		323.000000	
Customer#000144617	144617	3043270	1997-02-12
00:00:00.000 530604.440000		317.000000	
Customer#000013940	13940	2232932	1997-04-13
00:00:00.000 522720.610000		304.000000	
Customer#000066790	66790	2199712	1996-09-30
00:00:00.000 515531.820000		327.000000	



Customer#000046435	46435	4745607	1997-07-03	00:00:00.000 447606.650000	320.000000	
00:00:00.000 508047.990000		309.000000		Customer#000088876	88876	1993-12-30
Customer#000015272	15272	3883783	1993-07-28	00:00:00.000 446717.460000	304.000000	
00:00:00.000 500241.330000		302.000000		Customer#000036619	36619	1995-01-17
Customer#000146608	146608	3342468	1994-06-12	00:00:00.000 446704.090000	328.000000	
00:00:00.000 499794.580000		303.000000		Customer#000141823	141823	1996-12-29
Customer#000096103	96103	5984582	1992-03-16	00:00:00.000 446269.120000	310.000000	
00:00:00.000 494398.790000		312.000000		Customer#000053029	53029	1993-08-13
Customer#000024341	24341	1474818	1992-11-15	00:00:00.000 446144.490000	302.000000	
00:00:00.000 491348.260000		302.000000		Customer#000018188	18188	1995-01-25
Customer#000137446	137446	5489475	1997-05-23	00:00:00.000 443807.220000	308.000000	
00:00:00.000 487763.250000		311.000000		Customer#000066533	66533	1995-10-21
Customer#000107590	107590	4267751	1994-11-04	00:00:00.000 443576.500000	305.000000	
00:00:00.000 485141.380000		301.000000		Customer#000037729	37729	1995-06-29
Customer#000050008	50008	2366755	1996-12-09	00:00:00.000 441082.970000	309.000000	
00:00:00.000 483891.260000		302.000000		Customer#000003566	3566	1998-01-04
Customer#000015619	15619	3767271	1996-08-07	00:00:00.000 439803.360000	304.000000	
00:00:00.000 480083.960000		318.000000		Customer#000045538	45538	1994-05-22
Customer#000077260	77260	1436544	1992-09-12	00:00:00.000 436275.310000	305.000000	
00:00:00.000 479499.430000		307.000000		Customer#000081581	81581	1995-11-04
Customer#000109379	109379	5746311	1996-10-10	00:00:00.000 435405.900000	305.000000	
00:00:00.000 478064.110000		302.000000		Customer#000119989	119989	1997-09-20
Customer#000054602	54602	5832321	1997-02-09	00:00:00.000 434568.250000	320.000000	
00:00:00.000 471220.080000		307.000000		Customer#000003680	3680	1998-07-03
Customer#000105995	105995	2096705	1994-07-03	00:00:00.000 433525.970000	301.000000	
00:00:00.000 469692.580000		307.000000		Customer#000113131	113131	1995-12-15
Customer#000148885	148885	2942469	1992-05-31	00:00:00.000 432957.750000	301.000000	
00:00:00.000 469630.440000		313.000000		Customer#000141098	141098	1995-09-24
Customer#000114586	114586	551136	1993-05-19	00:00:00.000 430986.690000	301.000000	
00:00:00.000 469605.590000		308.000000		Customer#000093392	93392	1997-01-22
Customer#000105260	105260	5296167	1996-09-06	00:00:00.000 425487.510000	304.000000	
00:00:00.000 469360.570000		303.000000		Customer#000015631	15631	1994-05-12
Customer#000147197	147197	1263015	1997-02-02	00:00:00.000 419879.590000	302.000000	
00:00:00.000 467149.670000		320.000000		Customer#000112987	112987	1996-09-17
Customer#000064483	64483	2745894	1996-07-04	00:00:00.000 418161.490000	305.000000	
00:00:00.000 466991.350000		304.000000		Customer#000012599	12599	1998-02-12
Customer#000136573	136573	2761378	1996-05-31	00:00:00.000 415200.610000	304.000000	
00:00:00.000 461282.730000		301.000000		Customer#000105410	105410	1996-03-05
Customer#000016384	16384	502886	1994-04-12	00:00:00.000 412754.510000	302.000000	
00:00:00.000 458378.920000		312.000000		Customer#000149842	149842	1994-05-30
Customer#000117919	117919	2869152	1996-06-20	00:00:00.000 411329.350000	302.000000	
00:00:00.000 456815.920000		317.000000		Customer#000010129	10129	1994-03-21
Customer#000012251	12251	735366	1993-11-24	00:00:00.000 409129.850000	309.000000	
00:00:00.000 455107.260000		309.000000		Customer#000069904	69904	1996-10-19
Customer#000120098	120098	1971680	1995-06-14	00:00:00.000 408513.000000	305.000000	
00:00:00.000 453451.230000		308.000000		Customer#000017746	17746	1997-04-09
Customer#000066098	66098	5007490	1992-08-07	00:00:00.000 408446.930000	303.000000	
00:00:00.000 453436.160000		304.000000		Customer#000013072	13072	1998-03-15
Customer#000117076	117076	4290656	1997-02-05	00:00:00.000 399195.470000	301.000000	
00:00:00.000 449545.850000		301.000000		Customer#000082441	82441	1994-02-07
Customer#000129379	129379	4720454	1997-06-07	00:00:00.000 382579.740000	305.000000	
00:00:00.000 448665.790000		303.000000		Customer#000088703	88703	1994-01-30
Customer#000126865	126865	4702759	1994-11-07	00:00:00.000 363812.120000	302.000000	

(57 row(s) affected)

QUERY 19 FOR VALIDATION

-- using default substitutions

/\* TPC\_H Query 19 - Discounted Revenue \*/

```

SELECT SUM(L_EXTENDEDPRICE* (1 - L_DISCOUNT))AS REVENUE
FROMLINEITEM,
PART
WHERE (P_PARTKEY = L_PARTKEY AND
P_BRAND = 'Brand#12' AND
P_CONTAINER IN ( 'SM CASE', 'SM BOX', 'SM PACK', 'SM PKG') AND
L_QUANTITY >= 1 AND
L_QUANTITY <= 1 + 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 >= 10 AND
L_QUANTITY <= 10 + 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#34' AND
P_CONTAINER IN ( 'LG CASE', 'LG BOX', 'LG PACK', 'LG PKG') AND
L_QUANTITY >= 20 AND
L_QUANTITY <= 20 + 10 AND
P_SIZE BETWEEN 1 AND 15 AND
L_SHIPMODE IN ('AIR', 'AIR REG') AND
L_SHIPINSTRUCT= 'DELIVER IN PERSON'
)
REVENUE
-----
3083843.057800

```

(1 row(s) affected)

QUERY 20 FOR VALIDATION

-- using default substitutions

/\* 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
FROM PART
WHERE P_NAME like 'forest%'
) AND
PS_AVAILQTY > (SELECT 0.5 * sum(L_QUANTITY)
FROM LINEITEM
WHERE 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 = 'CANADA'
ORDER BY S_NAME

```

S_NAME	S_ADDRESS
Supplier#000000020	iybAE, RmTymrZVYaFZva2SH, j
Supplier#000000091	YV45D7TkfdQanOOZ7q9QxkyGUapU1oOWU6q3
Supplier#000000197	YC2Acon6kjY3zj3Fbxs2k4VdF7X0cd2F
Supplier#000000226	83qOdU2EYRdPQAQhEtn GRZEd
Supplier#000000285	Br7e1nnt1yxrw6ImgpJ7YdhFDjuBf
Supplier#000000378	FfbhyCxWvcPrO8ltp9
Supplier#000000402	i9Sw4DoyMhzhKXCH9By, AYSgmD
Supplier#000000530	0qwCMwobKY OcmLyfRXlagA8ukENJv,
Supplier#000000688	D fw5ocppmZpYBBIPI718hCihLDZ5KhKX
Supplier#000000710	f19YPvOyb QoYwjKC, oPycpGfieBACwKJo
Supplier#000000736	l6i2nMwVuovfKnuVgaSGK2rDy65DlAFLegiL7
Supplier#000000761	z1SLelQUj2XrvTTFnv7WAcYZGvVMtX882d4
Supplier#000000884	bmhEShejaS
Supplier#000000887	urEaTejh5POADP2ARrf
Supplier#000000935	ij98czM 2KzWe7dTOxB8sq0UfCdvrx
Supplier#000000975	, AC e, tBpNwKb5xMUzeohxlRn, hdzJo73gFQF8y
Supplier#000001263	rQWr6nf8ZhB2TAiDIvo5Io
Supplier#000001399	LmrocnIMSyYOWuAnX7
Supplier#000001446	lch9HMNU1R7a0LIybsUodVknk6
Supplier#000001454	TOpimgu2TVXIjhiL93h,
Supplier#000001500	wDmF5xLxtQch9ctVu,
Supplier#000001602	uKNWTeafaM644
Supplier#000001626	UhxNRzUuldTfmp0
Supplier#000001682	pXTkGxrTQVyH1Rr
Supplier#000001699	Q9C4rfJ26oijVPqqcqVXeRI
Supplier#000001700	7hMlCof1Y5zLFg
Supplier#000001726	TeRY7TtTH24sEword7yAaSkjx8
Supplier#000001730	Rc8e, lPybn r6zo0VJIEiD0UD vkh

Supplier#000001746 qWsendlOekQG1aW4uq06uQaCm5lse8lirv7 hBRd  
Supplier#000001752 Fra7outx41THYJaRThdOGiBk  
Supplier#000001856 jXcRgzYF0ah05iR8p6w5SbJLcUGyYiURPvFwUWM  
Supplier#000001931 FpJbMU2h6ZR2eBv8I9NIxF  
Supplier#000001939 Nrk, JA4bfReUs  
Supplier#000001990 DSDJkCgBJzuPg1yuM, CUDLnsRliOxkkHezTCA  
Supplier#000002020 jB6r1d7MxP6co  
Supplier#000002022 dwebGX7Id2pc25YvY33  
Supplier#000002036 20ytTtVobjKUII2WCB0A  
Supplier#000002204 uYmlr46C06udCqanj0KiRsoTQakZsEyssL  
Supplier#000002243 nSOEV3JeOU79  
Supplier#000002245 hz2qWXWVjOyKhqPYMoEwz6zFkrTaDM  
Supplier#000002282 ES21K9dxoW1I1TzWCj7ekdlNwSwnv1Z 6mQ, BKn  
Supplier#000002303 nCoWfpB6YOymbgOht7ltfklpkH1  
Supplier#000002373 RzHSxOTQmELCjxIBiVA52Z JB58rJhPRy1R  
Supplier#000002419 qydBQd14I515mVXa4fYY  
Supplier#000002481 nLKHUOn2M19TOA06Znq9GEMcIlMO2  
Supplier#000002571 JZUugz04c iJFLr1Gsz90 N,W 1rVHNIReyq  
Supplier#000002585 CsPoKpw2QuTY4AV1NkWuttneIa4SN  
Supplier#000002630 ZIQAvjNUY9KH5ive zm7k V1PiD17CCo21  
Supplier#000002719 4nnzQI2CbqREQUuTsXTBVUkaP4mNS3  
Supplier#000002721 HVdFAN2JHMQSpKm  
Supplier#000002730 lIFxR4fzm31C6, muzJw184z  
Supplier#000002775 yDcladaBD4ihH  
Supplier#000002853 rTNAOIItXka  
Supplier#000002875 6JgMi 9Qt6VmwL3Ltt1SRlKww0keLQ, RAZA  
Supplier#000002934 m, trBENywsARwg3DhB  
Supplier#000002941 Naddba 8YTEKekZyP0  
Supplier#000002960 KCPCESRGGo6vx8TygHh60nAYf9rStQT2T  
Supplier#000002980 B9k9yVsyaxVwktOSHezqHiAEp9id0SKzkw  
Supplier#000003062 LSQNgqY1xnOzz9zBCapy7HwOZQ  
Supplier#000003087 ANwe8QsZ4rgj1HSqVz991eWQ  
Supplier#000003089 s5b VCIZqMSZVa r g7LTdgc29GbTE7rI1x  
Supplier#000003095 HxON3jJhU13zjt, r mTD  
Supplier#000003201 E87yws6I, t0qNs4QW7UzExKiJnJDZwue  
Supplier#000003213 pxrRP4irQ1VoyfQ, dtf3  
Supplier#000003241 j06SU, LS9O3mwjAMOVIAneIhb  
Supplier#000003275 9x04nyJ2QJcX6vGf  
Supplier#000003288 EDdfNt7E5Uc, xLTupoIgyL4yY7ujh,  
Supplier#000003313 El2I7we, 049SPrvomUm4hZwJoOhZkvLxLJXgVH  
Supplier#000003314 jnisU8MzqO4iUB3zspcrysMw3DDUojS4q7LD  
Supplier#000003380 jPv0V, pszouFT3YsAq1P, kxT3u, gTFiEbRt, x  
Supplier#000003403 e3X2o , KCG9tsHji8A XXCxiF2hZWBw  
Supplier#000003421 Sh3dt9W5oeofFWovnFhrg,  
Supplier#000003441 zvfJIZs, oUuShHjpcX  
Supplier#000003590 sy79CMLxqb, Cbo  
Supplier#000003607 lNqFHQYjwSAkf  
Supplier#000003625 qY588W0Yk5iaUy1RXTgNrEKrMAjBYHcKs  
Supplier#000003656 eEYmmO2gmd JdfG32XtdGJV, db56  
Supplier#000003782 iVsPZg7bk06TqNmwi0LkBLurClzmrq  
Supplier#000003918 meRvRCsJoAbfqd0Re4  
Supplier#000003941 Pmb05mqfBMS61807WKqZJ 9vyv  
Supplier#000003994 W00LZp3NjK0

Supplier#000004005 V723FlwCy2eA4OgIu8TjBtOVUHP  
Supplier#000004033 ncsAhv9Je, kFXTNjfb2  
Supplier#000004140 0hL7DJyYjchL  
Supplier#000004165 wTJ2dZnQA8P2oi99N6DT47ndHy, XKD2  
Supplier#000004207 tF64pwiOM4IkWjN3mS, e06WuAjLx  
Supplier#000004236 dl, HPTJmGipxYsSqn9wmqkuWjst, mCeJ806T  
Supplier#000004246 Xha aXQF7u4qU3LsHD  
Supplier#000004278 bBdbbpBxIVp Di9  
Supplier#000004343 GK3sbopqrQEkWLMvVBFCG  
Supplier#000004346 S3076LEOwo  
Supplier#000004388 VfZ 11J, mwp4aS  
Supplier#000004406 Ah0ZaLu6VwufPWUz, 7kbXgYZhauEaHqGig  
Supplier#000004430 yvSsKNSTL5HLXBET4luOsPNLxKzAMk  
Supplier#000004522 xXtCKwsZDArxIBGdfzX2PgobGZsBg  
Supplier#000004527 p pVXCnxgcklWF6A1o3OHY3qW6  
Supplier#000004542 NJSbLJDroYG2y1r3rDiKg  
Supplier#000004574 1HvGwnVueZ5CIndc  
Supplier#000004655 67NqBc4 t3PG3F8aO IsqWNq4kGaPowYL  
Supplier#000004701 6jX4u47URzIMHf

+++++ Part of query output removed +++++

Supplier#000005192 JDp4rhXiDw0kf6RH  
Supplier#000005195 Woi3b2ZaicPh ZSfu1EfXhE  
Supplier#000005283 5fxYXwXy, TQX, MqDC2hxzyQ  
Supplier#000005300 gXG28YqpxU  
Supplier#000005386 Ub6AAfHpwLWP  
Supplier#000005426 9Dz2OVT1q sb4BK711jQ1XjPBYRPvO  
Supplier#000005484 saFdOR qW7AFY, 3asPqiiAa11Mo22pCoN0BtPrKo  
Supplier#000005505 d2sbjG43KwMPX  
Supplier#000005506 On f5ypzoWgB  
Supplier#000005516 XsN99Ks9wEvcohu6jRD2MeebQFf76mD8vovuY  
Supplier#000005536 Nzo9tGkpgbHT, EZ4D, 77MYKl4ah1C  
Supplier#000005605 7Vj6Eil0mThqkm  
Supplier#000005631 14TVrj1zo2SJEbYCDgpmwTlVwSqC  
Supplier#000005730 5rkb0PSews HvxkL8JaD41UpnSF2cg8H1  
Supplier#000005736 2dq XTYhtYWSfp  
Supplier#000005737 dmEWcS32C3kx, d, B95 OmYn48  
Supplier#000005797 , o, OebwRbSDmV19gN9fpWPCiqB Uogv1SR  
Supplier#000005836 tx3SjPD2ZuWGfBRH,  
Supplier#000005875 lK, sYiGzB94hSyHy9xvSZFbVQNCZe2LXZuGbs  
Supplier#000005974 REhR5jE, lLusQXvf54SwYySgsSSVFhu  
Supplier#000005989 rjFY, 5kgLpBu7c  
Supplier#000006059 4m0cv8MwJ9yX2v1wI Z  
Supplier#000006065 UiI2Cy3W4Tu5sLk LuvXLRy6KihlGv  
Supplier#000006070 TalC5m0pDrO6DZbngfmGmqe  
Supplier#000006109 rY5gbfh3dKHnlycQUTPGCwmbe  
Supplier#000006121 S92ycWwEzYyW4GspCBJN1WmuHhoZ  
Supplier#000006215 j2iEbTsl, 5PWdqWZ7klyiIsb7qtiiZlDIPeO  
Supplier#000006217 RVN23SYT9jenUeaWGxUd  
Supplier#000006274 S3yTZTqXtKUq g QQgcW9 AqhCkNZsW51hHuwU

Supplier#000006435 xIge69XszYbnO4Eon7cHHO8y  
Supplier#000006463 7 wkDj2EO49iotley2kmIM AdpLSszGV3RNWj  
Supplier#000006493 oJv f, sNaB6Hm7r, fknDVTL63raJgAjZK  
Supplier#000006521 b9 2zjHzxR  
Supplier#000006607 3F 2e2ggD5u5B  
Supplier#000006706 Ak4ga, ePulQZ6C3qkrqjosaX0gxvqS9vkbe  
Supplier#000006761 n4jhxGMqB5prD1HhpLvwrWstOLllla  
Supplier#000006808 HGd2Xo 9nEcHJhZvXjXxWKIpaT  
Supplier#000006858 fnlINT885vBBhsWwTGiZ0o22thwGY16h GHJj21  
Supplier#000006872 XIDPIA7PLXCWK6SeEclD  
Supplier#000006949 mLxYUJhsGcLtKe ,GFirNu183AvT  
Supplier#000006985 PrUUiboQpy, OtgJ01Z4BxJQYyrw9c3I  
Supplier#000007072 2tRyX9M1a 4Rcm57s779F1ANG9jlpK  
Supplier#000007098 G3j8g0KC4OcbAu2OVOPhrXQWMCUdjQ8wgCHOExu  
Supplier#000007135 ls DoKV7V5ulfQy9V  
Supplier#000007160 TqDGBULB3cTqIT6FKDvm9BS4e4v, zWyQPb  
Supplier#000007169 tEc95D2moN9S84nd550, dlnW  
Supplier#000007322 wr7dgte5q MAJiY0uwm13MyDkSMX1  
Supplier#000007365 51xhROLvQMj05DndtZWt  
Supplier#000007398 V8eE6oZ00OFNU,  
Supplier#000007402 4UVv58erylrmqSR5  
Supplier#000007448 yhhpWiJi7EJ6Q5VCAQ  
Supplier#000007477 9m9jOwfhWzCvVHxkU, PpAaxwSH0h  
Supplier#000007509 q8, V6LJRohJjHcOusG7aLTMg  
Supplier#000007561 rMcFg2530VC  
Supplier#000007789 rQ7cUcPrtudOyO3svNSkimqH6grfWT2Sz  
Supplier#000007801 69fi, U1r6enUb  
Supplier#000007818 yhhc2CQec Jrvc8zqBi83  
Supplier#000007885 u3sicchh5ZpyTUpN1cJKncAoabIWgY  
Supplier#000007918 r, v9mBQ6LoEYyjl  
Supplier#000007926 ErzCF80K9Uy  
Supplier#000007957 ELwnio14ssoU1 dRyZIL OK3Vtzb  
Supplier#000007965 F7Un51J7p5hhj  
Supplier#000007968 DsF9U1Z2Fo6HXN9aErvyg1ikHoD582HSGZpP  
Supplier#000007998 LnASFBfYRF0o9d6d, asBvVq9Lo2P  
Supplier#000008168 aOa82a8ZbKcnfDLX  
Supplier#000008231 IK7eGw Yj90sTdpSP, vcqWxLB  
Supplier#000008243 2AyePMkDqmqzVzjGTizXthFLO8h EiudCMxOmIIG  
Supplier#000008275 B1bNdfWg, gpXKQ1LN  
Supplier#000008323 75I18sZmASwm POeherMdj9tmpyeQ, BfCXN5BIAB  
Supplier#000008366 h778cEj14BuW9OEKlvPTWq4iWASR6EBBXN7zeS8  
Supplier#000008423 RQhKnkAhRODAr3Ix4Q1weMMn00hNe Kq  
Supplier#000008480 4sSDA4ACReklNjEm5T6b  
Supplier#000008532 Uc29q4, 5xVdDOF87UZrchr4xWS0ihEUXuh  
Supplier#000008595 MH0iB73GQ3z UW3O DbCbqmc  
Supplier#000008610 SgVgP90vP452sUNTgzL9zKwXHXAZV6tV  
Supplier#000008705 aE, trRNdPx, 4yinTD9O3DebDIP  
Supplier#000008742 HmPlQEzKCPecTUL14, kKq  
Supplier#000008841 I 85Lu1sekbq2xrSIzm0  
Supplier#000008895 2cH4okfaLSZTTg8sKRbbJQxkmeFu2Esj  
Supplier#000008967 2kwEHYMG 7FwozNImAUE6mH0hYtqYculJM  
Supplier#000008972 w2vF6 D5YZO3visPXsqVfLADTK  
Supplier#000009032 qK, trB6Sdy4Dz1BRUFNy

Supplier#000009147 rOAuryHxpZ9eOvx  
Supplier#000009252 F7cZaPUHwh1 ZKyj3xmAVWC1XdP ue1p5m,i  
Supplier#000009278 RqYtZgxj93CLX OmcYfCENoefD  
Supplier#000009327 uoqMdf7e7Gj9dbQ53  
Supplier#000009430 igRqmneFt  
Supplier#000009567 r4Wfx4c3xsEAjCgJ71HHZByornl D9vrztXlv4  
Supplier#000009601 51m637bO, Rw5DnHWFUvLacRx9  
Supplier#000009709 rRnCbHYgDg19PZYnyWKVYSUW0vKg  
Supplier#000009753 wLhVEcRmd7PkJF4FBnGK7Z  
Supplier#000009796 z, y4Idmr15DOvPUqYG  
Supplier#000009799 4wNjXGa4OKW1  
Supplier#000009811 E3iuyq7UnZxU7oPZIE2Gu6  
Supplier#000009812 APFRMy3lCbGfGa53n5t9DxzFPQPgnjrGt32  
Supplier#000009862 rJzweWeN58  
Supplier#000009868 ROjGgx5gvtkmnUUoeyy7v  
Supplier#000009869 ucLqxzrpBTRMewGSM29t0rNTM30g1Tu3Xgg3mKag  
Supplier#000009899 7XdpAhrzrlt, UQFZE  
Supplier#000009974 7wJ, J5DKcxSU4KplcQLpbcAvB5AsvKT

(204 row(s) affected)

**QUERY 21 FOR VALIDATION**

-----  
-- using default substitutions

/\* TPC\_H Query 21 - Suppliers Who Kept Orders Waiting \*/

```

SELECT TOP 100
S_NAME,
COUNT(*) AS NUMWAIT
FROMSUPPLIER,
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 *
FROMLINEITEM L2
WHERE L2.L_ORDERKEY = L1.L_ORDERKEY AND
L2.L_SUPPKEY <> L1.L_SUPPKEY
) AND
NOT EXISTS (SELECT *
FROMLINEITEM 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 = 'SAUDI ARABIA'
GROUP BY S_NAME
ORDER BY NUMWAIT DESC,

```

S_NAME	NUMWAIT
Supplier#000002829	20
Supplier#000005808	18
Supplier#000000262	17
Supplier#000000496	17
Supplier#000002160	17
Supplier#000002301	17
Supplier#000002540	17
Supplier#000003063	17
Supplier#000005178	17
Supplier#000008331	17
Supplier#000002005	16
Supplier#000002095	16
Supplier#000005799	16
Supplier#000005842	16
Supplier#000006450	16
Supplier#000006939	16
Supplier#000009200	16
Supplier#000009727	16
Supplier#000000486	15
Supplier#000000565	15
Supplier#000001046	15
Supplier#000001047	15
Supplier#000001161	15
Supplier#000001336	15
Supplier#000001435	15
Supplier#000003075	15
Supplier#000003335	15
Supplier#000005649	15
Supplier#000006027	15
Supplier#000006795	15
Supplier#000006800	15
Supplier#000006824	15
Supplier#000007131	15
Supplier#000007382	15
Supplier#000008913	15
Supplier#000009787	15
Supplier#000000633	14
Supplier#000001960	14
Supplier#000002323	14
Supplier#000002490	14
Supplier#000002993	14
Supplier#000003101	14
Supplier#000004489	14
Supplier#000005435	14
Supplier#000005583	14
Supplier#000005774	14
Supplier#000007579	14
Supplier#000008180	14
Supplier#000008695	14

Supplier#000009224	14
Supplier#000000357	13
Supplier#000000436	13
Supplier#000000610	13
Supplier#000000788	13
Supplier#000000889	13
Supplier#000001062	13
Supplier#000001498	13
Supplier#000002056	13
Supplier#000002312	13
Supplier#000002344	13
Supplier#000002596	13
Supplier#000002615	13
Supplier#000002978	13
Supplier#000003048	13
Supplier#000003234	13
Supplier#000003727	13
Supplier#000003806	13
Supplier#000004472	13
Supplier#000005236	13
Supplier#000005906	13
Supplier#000006241	13
Supplier#000006326	13
Supplier#000006384	13
Supplier#000006394	13
Supplier#000006624	13
Supplier#000006629	13
Supplier#000006682	13
Supplier#000006737	13
Supplier#000006825	13
Supplier#000007021	13
Supplier#000007417	13
Supplier#000007497	13
Supplier#000007602	13
Supplier#000008134	13
Supplier#000008234	13
Supplier#000009435	13
Supplier#000009436	13
Supplier#000009564	13
Supplier#000009896	13
Supplier#000000379	12
Supplier#000000673	12
Supplier#000000762	12
Supplier#000000811	12
Supplier#000000821	12
Supplier#000001337	12
Supplier#000001916	12
Supplier#000001925	12
Supplier#000002039	12
Supplier#000002357	12
Supplier#000002483	12

(100 row(s) affected)

QUERY 22 FOR VALIDATION

-----  
-- using default substitutions

/\* 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 ('13', '31', '23', '29', '30', '18', '17') AND
C_ACCTBAL > (SELECT AVG(C_ACCTBAL)
FROM CUSTOMER
WHERE C_ACCTBAL > 0.00 AND
SUBSTRING(C_PHONE,1,2) IN ('13', '31', '23', '29', '30', '18', '17')
) AND
NOT EXISTS (SELECT *
FROM ORDERS
WHERE O_CUSTKEY = C_CUSTKEY
)
) AS CUSTSALE
GROUP BY CNTRYCODE
ORDER BY CNTRYCODE
```

```
CNTRYCODE NUMCUST TOTACCTBAL
-----
13          888      6737713.990000
17          861      6460573.720000
18          964      7236687.400000
23          892      6701457.950000
29          948      7158866.630000
30          909      6808436.130000
31          922      6806670.180000
```

(7 row(s) affected)

## APPENDIX D : SEED & QUERY SUBSTITUTION

STREAM00 SEED 420000118  
193  
225COPPERASIA  
3FURNITURE 1995-03-18  
41995-11-01  
5AFRICA 1994-01-01  
61994-01-01 0.0325  
7EGYPTMOROCCO  
8MOROCCO AFRICALARGE PLATED STEEL  
9moccasin  
101994-09-01  
11JORDAN0.0000003333  
12RAILTRUCK 1993-01-01  
13unusual packages  
141993-03-01  
151996-06-01  
16Brand#31 LARGE BURNISHED3327122529222832  
17Brand#13 JUMBO BOX  
18314  
19Brand#43 Brand#23 Brand#35 82023  
20medium 1994-01-01 JAPAN  
21ETHIOPIA  
2222242816111532

STREAM01 SEED 420000119  
1101  
212STEELAFRICA  
3MACHINERY 1995-03-04  
41993-08-01  
5AMERICA 1995-01-01  
61995-01-01 0.0824  
7VIETNAM GERMANY  
8GERMANY EUROPELARGE ANODIZED STEEL  
9maroon  
101993-06-01  
11ARGENTINA 0.0000003333  
12AIRTRUCK 1993-01-01  
13unusual packages  
141993-06-01  
151994-03-01  
16Brand#11 PROMO POLISHED361746626132333  
17Brand#15 JUMBO PACK  
18315  
19Brand#45 Brand#51 Brand#34 31030  
20turquoise 1997-01-01 BRAZIL  
21RUSSIA  
2223142520103419

STREAM02 SEED 420000120  
1109  
250NICKEL EUROPE  
3FURNITURE 1995-03-21  
41996-03-01  
5ASIA 1995-01-01  
61995-01-01 0.0625  
7JORDAN UNITED STATES  
8UNITED STATESAMERICAMEDIUM POLISHED STEEL  
9lawn  
101994-03-01  
11KENYA 0.0000003333  
12REG AIRTRUCK 1993-01-01  
13unusual packages  
141993-10-01  
151996-10-01  
16Brand#51 SMALL BRUSHED3919241825151337  
17Brand#12 JUMBO DRUM  
18313  
19Brand#42 Brand#44 Brand#33 91126  
20grey 1996-01-01 MOZAMBIQUE  
21MOROCCO  
2226111019302529

STREAM03 SEED 420000121  
1117  
238TINAFRICA  
3MACHINERY 1995-03-06  
41993-12-01  
5EUROPE 1995-01-01  
61995-01-01 0.0325  
7ETHIOPIA MOZAMBIQUE  
8MOZAMBIQUE AFRICAMEDIUM BURNISHED STEEL  
9hot  
101994-12-01  
11BRAZIL 0.0000003333  
12SHIPMAIL 1993-01-01  
13unusual packages  
141994-01-01  
151994-07-01  
16Brand#31 ECONOMY BURNISHED431629723331813  
17Brand#14 WRAP BOX  
18314  
19Brand#54 Brand#22 Brand#22 41223  
20royal 1994-01-01 FRANCE  
21GERMANY  
2234212918102317

STREAM04 SEED 420000122  
164  
226COPPER EUROPE  
3BUILDING 1995-03-23  
41996-07-01

5MIDDLE EAST 1995-01-01  
61995-01-01 0.0924  
7RUSSIA INDIA  
8INDIAASIASMALL BRUSHED COPPER  
9gainsboro  
101993-09-01  
11MOROCCO 0.0000003333  
12FOBMAIL1994-01-01  
13unusual packages  
141994-04-01  
151997-01-01  
16Brand#11 STANDARD PLATED1611494323132719  
17Brand#11 WRAP PACK  
18312  
19Brand#52 Brand#55 Brand#22 91330  
20cornsilk 1997-01-01 VIETNAM  
21UNITED STATES  
2221172910181124

STREAM05 SEED 420000123  
172  
214STEELAMERICA  
3MACHINERY 1995-03-08  
41994-04-01  
5AFRICA 1995-01-01  
61995-01-01 0.0624  
7KENYA ALGERIA  
8ALGERIA AFRICA SMALL PLATED COPPER  
9dodger  
101994-07-01  
11CANADA 0.0000003333  
12MAILRAIL1994-01-01  
13unusual requests  
141994-07-01  
151994-10-01  
16Brand#51 MEDIUM BRUSHED13229136121922  
17Brand#13 WRAP DRUM  
18313  
19Brand#54 Brand#43 Brand#21 41426  
20olive 1996-01-01 IRAQ  
21MOZAMBIQUE  
2221181213202615

STREAM06 SEED 420000124  
180  
21BRASS EUROPE  
3BUILDING 1995-03-25  
41996-11-01  
5ASIA 1996-01-01  
61996-01-01 0.0425  
7FRANCE PERU  
8PERUAMERICA SMALL ANODIZED COPPER  
9cornsilk

101993-04-01  
11MOZAMBIQUE 0.0000003333  
12RAILMAIL1994-01-01  
13unusual requests  
141994-10-01  
151997-05-01  
16Brand#31 PROMO ANODIZED12282320849627  
17Brand#15 SM BOX  
18315  
19Brand#11 Brand#21 Brand#15 101522  
20azure 1994-01-01 ALGERIA  
21INDIA  
2218293112171621



## APPENDIX E : REFRESH FUNCTION SOURCE CODE

### CreateRF1Proc.sql

```
-- File:          CREATERF1PROC.SQL
--              Microsoft TPC-H Benchmark Kit Ver. 2.00
--              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 than 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 LINEITEM 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. 2.00
--               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 than 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
END

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 LOAD_FG

create clustered index NEWLINEITEM_CLUIDX
on NEWLINEITEM (L_ORDERKEY)
on LOAD_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_ORDERDATE datetime not null,
 O_ORDERKEY bigint not null,
 O_CUSTKEY int not null,
 O_ORDERPRIORITY char(15) not null,
 O_SHIPPRIORITY int not null,
 O_CLERK char(15) not null,
 O_ORDERSTATUS char(1) not null,
 O_TOTALPRICE float not null,
 O_COMMENT varchar(79) not null) on LOAD_FG

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

```

```

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

create index MOD_OLDORDERS_IDX on MOD_OLDORDERS (O_ORDERKEY) on LOAD_FG

```

### 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 LOAD_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 LOAD_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 (FieldTerminator = '|' , RowTerminator
='|¥n',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 print.c for DBGEN

### print.c

```
/*
 * $Id: print.c,v 1.2 2005/01/03 20:08:59 jms Exp $
 *
 * Revision History
 * =====
 * $Log: print.c,v $
 * Revision 1.2 2005/01/03 20:08:59 jms
 * change line terminations
 *
 * Revision 1.1.1.1 2004/11/24 23:31:47 jms
 * re-establish external server
 *
 * Revision 1.4 2004/02/18 16:26:49 jms
 * 32/64 bit changes for overflow handling needed additional changes when ported
 back to windows
 *
 * Revision 1.3 2004/02/18 14:05:53 jms
 * porting changes for LINUX and 64 bit RNG
 *
 * Revision 1.2 2004/01/22 05:49:29 jms
 * AIX porting (AIX 5.1)
 *
 * Revision 1.1.1.1 2003/08/07 17:58:34 jms
 * recreation after CVS crash
 *
 * Revision 1.2 2003/08/07 17:58:34 jms
 * Convery RNG to 64bit space as preparation for new large scale RNG
 *
 * Revision 1.1.1.1 2003/04/03 18:54:21 jms
 * initial checkin
 *
 */
/* generate flat files for data load */
#include <stdio.h>
#ifdef VMS
#include <sys/types.h>
#endif
#ifdef SUN
#include <unistd.h>
#endif
#include <math.h>

#include "dss.h"
```

```
#include "dsstypes.h"
#include <string.h>

/*
 * Function Prototypes
 */
FILE *print_prep PROTO((int table, int update));
int pr_drange PROTO((int tbl, DSS_HUGE min, DSS_HUGE cnt, long num));

FILE *
print_prep(int table, int update)
{
    char upath[128];
    FILE *res;

    if (updates)
    {
        if (update > 0) /* updates */
        if (insert_segments)
        {
            int this_segment;
            if(strcmp(tdefs[table].name,"orders.tbl"))
                this_segment=++insert_orders_segment;
            else
                this_segment=++insert_lineitem_segment;
            sprintf(upath, "%s%c%s.u%d.%d",
                env_config(PATH_TAG, PATH_DFLT),
                PATH_SEP, tdefs[table].name, update%10000,this_segment);
        }
        else
        {
            sprintf(upath, "%s%c%s.u%d",
                env_config(PATH_TAG, PATH_DFLT),
                PATH_SEP, tdefs[table].name, update);
        }
        else /* deletes */
        if (delete_segments)
        {
            ++delete_segment;
            sprintf(upath, "%s%cdelete.u%d.%d",
                env_config(PATH_TAG, PATH_DFLT), PATH_SEP, -update%10000,
                delete_segment);
        }
        else
        {
            sprintf(upath, "%s%cdelete.%d",
                env_config(PATH_TAG, PATH_DFLT), PATH_SEP, -update);
        }
        return(fopen(upath, "w"));
    }
    res = tbl_open(table, "w");
    OPEN_CHECK(res, tdefs[table].name);
    return(res);
}
```

```

}
int
dbg_print(int format, FILE *target, void *data, int len, int sep)
{
    int dollars,
        cents;

    switch(format)
    {
    case DT_STR:
        if (columnar)
            fprintf(target, "%-*s", len, (char *)data);
        else
            fprintf(target, "%s", (char *)data);
        break;
    #ifdef MVS
    case DT_VSTR:
        /* note: only used in MVS, assumes columnar output */
        fprintf(target, "%c%c%-*s",
            (len >> 8) & 0xFF, len & 0xFF, len, (char *)data);
        break;
    #endif /* MVS */
    case DT_INT:
        if (columnar)
            fprintf(target, "%12ld", (long)data);
        else
            fprintf(target, "%ld", (long)data);
        break;
    case DT_HUGE:
        fprintf(target, HUGE_FORMAT, *(DSS_HUGE *)data);
        break;
    case DT_KEY:
        fprintf(target, "%ld", (long)data);
        break;
    case DT_MONEY:
        cents = (int)*(DSS_HUGE *)data;
        if (cents < 0)
        {
            fprintf(target, "-");
            cents = -cents;
        }
        dollars = cents / 100;
        cents %= 100;
        if (columnar)
            fprintf(target, "%12ld.%02ld", dollars, cents);
        else
            fprintf(target, "%ld.%02ld", dollars, cents);
        break;
    case DT_CHR:
        if (columnar)
            fprintf(target, "%c ", *(char *)data);
        else

```

```

        fprintf(target, "%c", *(char *)data);
        break;
    }
}
#ifdef EOL_HANDLING
    if (sep)
#endif /* EOL_HANDLING */
    if (!columnar)
        fprintf(target, "%c", SEPARATOR);

    return(0);
}

int
pr_cust(customer_t *c, int mode)
{
    static FILE *fp = NULL;

    if (fp == NULL)
        fp = print_prep(CUST, 0);

    PR_STRT(fp);
    PR_HUGE(fp, &c->custkey);
    PR_STR(fp, c->mktsegment, C_MSEG_LEN);
    PR_HUGE(fp, &c->nation_code);
    PR_VSTR(fp, c->name, C_NAME_LEN);
    PR_VSTR(fp, c->address,
        (columnar)?(long)(ceil(C_ADDR_LEN * V_STR_HGH)):c->alen);
    PR_STR(fp, c->phone, PHONE_LEN);
    PR_MONEY(fp, &c->acctbal);
    PR_VSTR_LAST(fp, c->comment,
        (columnar)?(long)(ceil(C_CMNT_LEN * V_STR_HGH)):c->clen);
    PR_END(fp);

    return(0);
}

/*
 * print the numbered order
 */
int
pr_order(order_t *o, int mode)
{
    static FILE *fp_o = NULL;
    static int last_mode = 0;

    if (fp_o == NULL || mode != last_mode)
    {
        if (fp_o)
            fclose(fp_o);
        fp_o = print_prep(ORDER, mode);
        last_mode = mode;
    }
}

```

```

PR_STRT(fp_o);
PR_STR(fp_o, o->odate, DATE_LEN);
PR_HUGE(fp_o, &o->okey);
PR_HUGE(fp_o, &o->custkey);
PR_STR(fp_o, o->opriority, O_OPRIO_LEN);
PR_INT(fp_o, o->spriority);
PR_STR(fp_o, o->clerk, O_CLRK_LEN);
PR_CHR(fp_o, &o->orderstatus);
PR_MONEY(fp_o, &o->totalprice);
PR_VSTR_LAST(fp_o, o->comment,
             (columnar)?(long)(ceil(O_CMNT_LEN * V_STR_HGH)):o->cflen);
PR_END(fp_o);

return(0);
}

/*
 * print an order's lineitems
 */
int
pr_line(order_t *o, int mode)
{
    static FILE *fp_l = NULL;
    static int last_mode = 0;
    long i;

    if (fp_l == NULL || mode != last_mode)
    {
        if (fp_l)
            fclose(fp_l);
        fp_l = print_prep(LINE, mode);
        last_mode = mode;
    }

    for (i = 0; i < o->lines; i++)
    {
        PR_STRT(fp_l);
        PR_STR(fp_l, o->l[i].sdate, DATE_LEN);
        PR_HUGE(fp_l, &o->l[i].okey);
        PR_MONEY(fp_l, &o->l[i].discount);
        PR_MONEY(fp_l, &o->l[i].eprice);
        PR_HUGE(fp_l, &o->l[i].suppkey);
        PR_HUGE(fp_l, &o->l[i].quantity);
        PR_CHR(fp_l, &o->l[i].rflag[0]);
        PR_HUGE(fp_l, &o->l[i].partkey);
        PR_CHR(fp_l, &o->l[i].lstatus[0]);
        PR_MONEY(fp_l, &o->l[i].tax);
        PR_STR(fp_l, o->l[i].cdate, DATE_LEN);
        PR_STR(fp_l, o->l[i].rdate, DATE_LEN);
        PR_STR(fp_l, o->l[i].shipmode, L_SMODE_LEN);
        PR_HUGE(fp_l, &o->l[i].lcnt);
        PR_STR(fp_l, o->l[i].shipinstruct, L_INST_LEN);
        PR_VSTR_LAST(fp_l, o->l[i].comment,

```

```

             (columnar)?(long)(ceil(L_CMNT_LEN * V_STR_HGH)):o->l[i].cflen);
        PR_END(fp_l);
    }

    return(0);
}

/*
 * print the numbered order *and* its associated lineitems
 */
int
pr_order_line(order_t *o, int mode)
{
    tdefs[ORDER].name = tdefs[ORDER_LINE].name;
    pr_order(o, mode);
    pr_line(o, mode);

    return(0);
}

/*
 * print the given part
 */
int
pr_part(part_t *part, int mode)
{
    static FILE *p_fp = NULL;

    if (p_fp == NULL)
        p_fp = print_prep(PART, 0);

    PR_STRT(p_fp);
    PR_HUGE(p_fp, &part->partkey);
    PR_VSTR(p_fp, part->type,
           (columnar)?(long)P_TYPE_LEN:part->tlen);
    PR_HUGE(p_fp, &part->size);
    PR_STR(p_fp, part->brand, P_BRND_LEN);
    PR_VSTR(p_fp, part->name,
           (columnar)?(long)P_NAME_LEN:part->nlen);
    PR_STR(p_fp, part->container, P_CNTR_LEN);
    PR_STR(p_fp, part->mfgr, P_MFG_LEN);
    PR_MONEY(p_fp, &part->retailprice);
    PR_VSTR_LAST(p_fp, part->comment,
                (columnar)?(long)(ceil(P_CMNT_LEN * V_STR_HGH)):part->cflen);
    PR_END(p_fp);

    return(0);
}

/*
 * print the given part's suppliers
 */
int

```



```

pr_psupp(part_t *part, int mode)
{
    static FILE *ps_fp = NULL;
    long        i;

    if (ps_fp == NULL)
        ps_fp = print_prep(PSUPP, mode);

    for (i = 0; i < SUPP_PER_PART; i++)
    {
        PR_STRT(ps_fp);
        PR_HUGE(ps_fp, &part->s[i].partkey);
        PR_HUGE(ps_fp, &part->s[i].suppkey);
        PR_MONEY(ps_fp, &part->s[i].scost);
        PR_HUGE(ps_fp, &part->s[i].qty);
        PR_VSTR_LAST(ps_fp, part->s[i].comment,
            (columnar)?(long)(ceil(PS_CMNT_LEN * V_STR_HGH)):part->s[i].clen);
        PR_END(ps_fp);
    }

    return(0);
}

/*
 * print the given part *and* its suppliers
 */
int
pr_part_psupp(part_t *part, int mode)
{
    tdefs[PART].name = tdefs[PART_PSUPP].name;
    pr_part(part, mode);
    pr_psupp(part, mode);

    return(0);
}

int
pr_supp(supplier_t *supp, int mode)
{
    static FILE *fp = NULL;

    if (fp == NULL)
        fp = print_prep(SUPP, mode);

    PR_STRT(fp);
    PR_HUGE(fp, &supp->suppkey);
    PR_HUGE(fp, &supp->nation_code);
    PR_STR(fp, supp->name, S_NAME_LEN);
    PR_VSTR(fp, supp->address,
        (columnar)?(long)(ceil(S_ADDR_LEN * V_STR_HGH)):supp->alen);
    PR_STR(fp, supp->phone, PHONE_LEN);
    PR_MONEY(fp, &supp->acctbal);
    PR_VSTR_LAST(fp, supp->comment,

```

```

        (columnar)?(long)(ceil(S_CMNT_LEN * V_STR_HGH)):supp->clen);
    PR_END(fp);

    return(0);
}

int
pr_nation(code_t *c, int mode)
{
    static FILE *fp = NULL;

    if (fp == NULL)
        fp = print_prep(NATION, mode);

    PR_STRT(fp);
    PR_HUGE(fp, &c->code);
    PR_STR(fp, c->text, NATION_LEN);
    PR_INT(fp, c->join);
    PR_VSTR_LAST(fp, c->comment,
        (columnar)?(long)(ceil(N_CMNT_LEN * V_STR_HGH)):c->clen);
    PR_END(fp);

    return(0);
}

int
pr_region(code_t *c, int mode)
{
    static FILE *fp = NULL;

    if (fp == NULL)
        fp = print_prep(REGION, mode);

    PR_STRT(fp);
    PR_HUGE(fp, &c->code);
    PR_STR(fp, c->text, REGION_LEN);
    PR_VSTR_LAST(fp, c->comment,
        (columnar)?(long)(ceil(R_CMNT_LEN * V_STR_HGH)):c->clen);
    PR_END(fp);

    return(0);
}

/*
 * NOTE: this routine does NOT use the BCD2_* routines. As a result,
 * it WILL fail if the keys being deleted exceed 32 bits. Since this
 * would require ~660 update iterations, this seems an acceptable
 * oversight
 */
int
pr_drange(int tbl, DSS_HUGE min, DSS_HUGE cnt, long num)
{
    static int last_num = 0;

```

```

static FILE *dfp = NULL;
DSS_HUGE child = -1;
DSS_HUGE start, last, new;

static DSS_HUGE rows_per_segment=0;
static DSS_HUGE rows_this_segment=0;

if (last_num != num)
{
    if (dfp)
        fclose(dfp);
    dfp = print_prep(tbl, -num);
    if (dfp == NULL)
        return(-1);
    last_num = num;
    rows_this_segment=0;
}

start = MK_SPARSE(min, num/ (10000 / refresh));
last = start - 1;
for (child=min; cnt > 0; child++, cnt--)
{
    new = MK_SPARSE(child, num/ (10000 / refresh));
    if (gen_rng == 1 && new - last == 1)
    {
        last = new;
        continue;
    }
}

if (gen_sql)
{
    fprintf(dfp,
"delete from %s where %s between %ld and %ld;\n",
tdefs[ORDER].name, "o_orderkey", start, last);
    fprintf(dfp,
"delete from %s where %s between %ld and %ld;\n",
tdefs[LINE].name, "l_orderkey", start, last);
    fprintf(dfp, "commit work;\n");
}
else
    if (gen_rng)
    {
        PR_STRT(dfp);
        PR_HUGE(dfp, &start);
        PR_HUGE(dfp, &last);
        PR_END(dfp);
    }
    else
    {
        if (delete_segments)
        {
            if (rows_per_segment==0)
                rows_per_segment = (cnt / delete_segments) + 1;

```

```

if(++rows_this_segment) > rows_per_segment)
{
    fclose(dfp);
    dfp = print_prep(tbl, -num);
    if (dfp == NULL) return(-1);
    last_num = num;
    rows_this_segment=1;
}

PR_STRT(dfp);
PR_HUGE(dfp, &new);
PR_END(dfp);
}

start = new;
last = new;
}
if (gen_rng)
{
    PR_STRT(dfp);
    PR_HUGE(dfp, &start);
    PR_HUGE(dfp, &last);
    PR_END(dfp);
}

return(0);
}

/*
 * verify functions: routines which replace the pr_routines and generate a
pseudo checksum
 * instead of generating the actual contents of the tables. Meant to allow
large scale data
 * validation without requiring a large amount of storage
 */
int
vrf_cust(customer_t *c, int mode)
{
    VRF_STRT(CUST);
    VRF_INT(CUST, c->custkey);
    VRF_STR(CUST, c->name);
    VRF_STR(CUST, c->address);
    VRF_INT(CUST, c->nation_code);
    VRF_STR(CUST, c->phone);
    VRF_MONEY(CUST, c->sacctbal);
    VRF_STR(CUST, c->mktsegment);
    VRF_STR(CUST, c->comment);
    VRF_END(CUST);

    return(0);
}

/*
 * print the numbered order

```

```

*/
int
vrf_order(order_t *o, int mode)
{
    VRF_STRT(ORDER);
    VRF_HUGE(ORDER, o->okey);
    VRF_INT(ORDER, o->custkey);
    VRF_CHR(ORDER, o->orderstatus);
    VRF_MONEY(ORDER, o->totalprice);
    VRF_STR(ORDER, o->odate);
    VRF_STR(ORDER, o->opriority);
    VRF_STR(ORDER, o->clerk);
    VRF_INT(ORDER, o->spriority);
    VRF_STR(ORDER, o->comment);
    VRF_END(ORDER);

    return(0);
}

/*
 * print an order's lineitems
 */
int
vrf_line(order_t *o, int mode)
{
    int i;

    for (i = 0; i < o->lines; i++)
    {
        VRF_STRT(LINE);
        VRF_HUGE(LINE, o->l[i].okey);
        VRF_INT(LINE, o->l[i].partkey);
        VRF_INT(LINE, o->l[i].suppkey);
        VRF_INT(LINE, o->l[i].lcnt);
        VRF_INT(LINE, o->l[i].quantity);
        VRF_MONEY(LINE, o->l[i].eprice);
        VRF_MONEY(LINE, o->l[i].discount);
        VRF_MONEY(LINE, o->l[i].tax);
        VRF_CHR(LINE, o->l[i].rflag[0]);
        VRF_CHR(LINE, o->l[i].lstatus[0]);
        VRF_STR(LINE, o->l[i].sdate);
        VRF_STR(LINE, o->l[i].cdate);
        VRF_STR(LINE, o->l[i].rdate);
        VRF_STR(LINE, o->l[i].shipinstruct);
        VRF_STR(LINE, o->l[i].shipmode);
        VRF_STR(LINE, o->l[i].comment);
        VRF_END(LINE);
    }

    return(0);
}

/*
 * print the numbered order *and* its associated lineitems
 */
int
vrf_order_line(order_t *o, int mode)
{
    vrf_order(o, mode);
    vrf_line(o, mode);

    return(0);
}

/*
 * print the given part
 */
int
vrf_part(part_t *part, int mode)
{
    VRF_STRT(PART);
    VRF_INT(PART, part->partkey);
    VRF_STR(PART, part->name);
    VRF_STR(PART, part->mfgr);
    VRF_STR(PART, part->brand);
    VRF_STR(PART, part->type);
    VRF_INT(PART, part->size);
    VRF_STR(PART, part->container);
    VRF_MONEY(PART, part->retailprice);
    VRF_STR(PART, part->comment);
    VRF_END(PART);

    return(0);
}

/*
 * print the given part's suppliers
 */
int
vrf_psupp(part_t *part, int mode)
{
    long i;

    for (i = 0; i < SUPP_PER_PART; i++)
    {
        VRF_STRT(PSUPP);
        VRF_INT(PSUPP, part->s[i].partkey);
        VRF_INT(PSUPP, part->s[i].suppkey);
        VRF_INT(PSUPP, part->s[i].qty);
        VRF_MONEY(PSUPP, part->s[i].scost);
        VRF_STR(PSUPP, part->s[i].comment);
        VRF_END(PSUPP);
    }

    return(0);
}

```

```

}
}

/*
 * print the given part *and* its suppliers
 */
int
vrf_part_psupp(part_t *part, int mode)
{
    vrf_part(part, mode);
    vrf_psupp(part, mode);

    return(0);
}

int
vrf_supp(supplier_t *supp, int mode)
{
    VRF_STRT(SUPP);
    VRF_INT(SUPP, supp->suppkey);
    VRF_STR(SUPP, supp->name);
    VRF_STR(SUPP, supp->address);
    VRF_INT(SUPP, supp->nation_code);
    VRF_STR(SUPP, supp->phone);
    VRF_MONEY(SUPP, supp->acctbal);
    VRF_STR(SUPP, supp->comment);
    VRF_END(SUPP);

    return(0);
}

int
vrf_nation(code_t *c, int mode)
{
    VRF_STRT(NATION);
    VRF_INT(NATION, c->code);
    VRF_STR(NATION, c->text);
    VRF_INT(NATION, c->join);
    VRF_STR(NATION, c->comment);
    VRF_END(NATION);

    return(0);
}

int
vrf_region(code_t *c, int mode)
{
    VRF_STRT(REGION);
    VRF_INT(REGION, c->code);
    VRF_STR(REGION, c->text);
    VRF_STR(REGION, c->comment);
    VRF_END(fp);

    return(0);
}

```

F.1.1.2 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 qryCons 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
    End For
End Function

```

```

        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

GetParameterValueErr:

```

```

    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

End Function

```



```

' 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
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 mcarrConnDtIs.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 mcarrConnDtIs.Count - 1
        If mcarrConnDtIs(lngIndex).ConnNameId = lConnNameId And _
            mcarrConnDtIs(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 = mcarrConnDtIs(lngQueryElement)
End Function
Public Property Get Count() As Long

    Count = mcarrConnDtIs.Count
End Property
Public Property Get Item(lngIndex As Long) As cConnDtl

```

```

Attribute Item.VB_UserMemId = 0

    Set Item = mcarrConnDtIs(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 mcarrConnDtIs.Count - 1
        Set cTempParam = mcarrConnDtIs(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 mcarrConnDtIs.Count - 1
        Set cTempParam = mcarrConnDtIs(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 mcarrConnDtIs = 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
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrModuleName & "AssignParameters", _
        LoadResString(errAssignParametersFailed)
Case "[quoted_id]"
    prmParam.Value = QuotedIdentifiers
Case "[a_nulls]"
    prmParam.Value = AnsiNulls
Case "[show_gp]"
    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

Private Sub Class_Initialize()

    Set mcarrConnections = New cNodeCollections

End Sub

Private Sub Class_Terminate()

    Set mcarrConnections = Nothing

End Sub

```

```

        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 mdbaConstraintDB 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 = mdbaConstraintDB.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
' mdbaConstraintDB.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 = mdbaConstraintDB.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
' mdbaConstraintDB.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 mdbaConstraintDB = vdata

End Property

Public Property Get NodeDB() As Database

    Set NodeDB = mdbaConstraintDB

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
        ' Propagate 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
        ' Propagate 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

```

```

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 cRunCollt

' 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)
        End If
    Next 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

```

```

' 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)
        End If
    Next 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."
        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
End 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
        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
End 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
        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
End Sub

```



```

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."
    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
```

### **cInstances.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
    End If

QueryInstanceErr:
    Exit Function
End Function
```

```
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:
    Exit Function
End Function
```

```

' 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 lngIndex As Long

    InstanceAborted = False

    For lngIndex = 0 To Count() - 1
        If mcInstances(lngIndex).Step.StepId = cSubStepRec.StepId And _
            mcInstances(lngIndex).Status = gintAborted Then
            InstanceAborted = True
            Exit For
        End If
    Next lngIndex

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 mdbaIteratorDB 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 = mdfsIteratorDB.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 mdfsIteratorDB 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 = mdfsIteratorDB.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 = mdfsIteratorDB.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 mdsIteratorDB = vdata
    End Property

    Public Property Get NodeDB() As Database
        Set NodeDB = mdsIteratorDB
    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 mdfsIteratorDB = RHS
End Property

Private Property Get cNode_NodeDB() As DAO.Database

    Set cNode_NodeDB = mdfsIteratorDB

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

```

```

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 mdbaNodeDb 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-existent 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 = mdbaNodeDb

' 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 = "mmddy"

' 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)
            End If
        End If
    Loop

```

- 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 character 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:      ConnectionCommon.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 mdbaStepMaster 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 = mdbaStepMaster
    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 mdbaStepMaster = vdata
End Property
Public Property Get NodeDB() As Database
    Set NodeDB = mdbaStepMaster
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 = mdbaStepMaster.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 mdbaStepMaster 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
        ' Propagate 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 = mdfsStepMaster.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 = mdbStepMaster
    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 mdbStepMaster = 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 RunConnDtIs As cConnDtIs
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 msAbortDtIs 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

```



```

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

```



```

    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

```

```

    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

```

```

    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
    ' decrement 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
' decrement 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
End Sub

```

```

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 StepDtIs 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
    End Sub

```

```

        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
'   End 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 WspPostExecute() As Variant
    WspPostExecute = mcvntWspPostCons
End Property
Public Property Let WspPostExecute(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,
Elapsed:=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,
Elapsed:=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,
Elapsed:=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, lngInstanceId As Long, Status As InstanceStatus)
    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep17.Index, lngInstanceId,
        Status)
End Sub

Private Sub cExecStep17_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, lngInstanceId As Long)
    Call TimeStartUpdateForStep(cStepRecord, lngInstanceId, dtmStartTime)
End Sub

Private Sub cExecStep18_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)
    Call TimeUpdateForProcess(cStepRecord, lngInstanceId, 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, lngInstanceId As Long, Status As InstanceStatus)
    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep18.Index, lngInstanceId,
        Status)
End Sub

Private Sub cExecStep18_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, lngInstanceId As Long)
    Call TimeStartUpdateForStep(cStepRecord, lngInstanceId, dtmStartTime)
End Sub

Private Sub cExecStep19_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)
    Call TimeUpdateForProcess(cStepRecord, lngInstanceId, 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, lngInstanceId As Long, Status As InstanceStatus)
    Call StepTerminated(cStepRecord, dtmEndTime, cExecStep19.Index, lngInstanceId,
        Status)
End Sub

Private Sub cExecStep19_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, lngInstanceId As Long)
    Call TimeStartUpdateForStep(cStepRecord, lngInstanceId, dtmStartTime)
End Sub

Private Sub cExecStep20_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)
    Call TimeUpdateForProcess(cStepRecord, lngInstanceId, 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, EndTime:=dtmEndTime,
    Command:=strCommand)

```

```

    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,
Elapsed:=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,
Elapsed:=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,
Elapsed:=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, Command:=strCommand)
End Sub

```

```

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

```



```

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

```

```

    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)

```

```

    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

```

```

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 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

```

```

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, Command:=strCommand,
StartTime:=dtmStartTime)
End Sub

```

```

    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, Command:=strCommand,
StartTime:=dtmStartTime)
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 = gintExecutesShell 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 = gintExecutesShell 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
End Function

```

```

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
            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

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 mdbaLoadDb 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 = mdbaLoadDb.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 & " ) "

mdbaLoadDb.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 = mdbaLoadDb.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 = mdbaLoadDb.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 = mdsLoadDb.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 = mdfsLoadDb.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 mcRunConnDtIs = New cConnDtIs
    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 mcRunConnDtIs = 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)

End Sub

```

```

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"
    mdsDatabase.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 mdfsDatabase 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

        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 = mdfsDatabase

' 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 mdfsDatabase = vdata
  Set mcIterators.NodeDB = vdata

End Property

Public Property Get NodeDB() As Database

  Set NodeDB = mdfsDatabase

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
    mblsNewVersion = 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 = mdbaDatabase.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 = mdbaDatabase.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 <> " & mFldValue.MakeStringFieldValid(mstrVersionNo)

BugMessage sUpdate
mObsDatabase.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 = mdba.Database.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
mdba.Database.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(qstrEmptyString, 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 StepTypeError

    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

StepTypeError:
    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 =
    On Error GoTo 0
    Err.Raise vbObjectError + errGetStepTypeFailed, _
        mstrSource, _
        LoadResString(errGetStepTypeFailed)
End Property

Public Property Let StepType(vdata As gintStepType)

    On Error GoTo StepTypeError

    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

StepTypeError:
    LogErrors Errors
    mstrSource = mstrModuleName & "StepType"
    On Error GoTo 0
    Err.Raise vbObjectError + errLetStepTypeFailed, _
        mstrSource, _
        LoadResString(errLetStepTypeFailed)
End Property

```

```

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 = mdbDatabase
    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)
    End While

```

```

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)
                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 mdbStepMaster 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 mdbStepMaster = 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 = mdsStepMaster.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 = mdsStepMaster.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
        ' Propagate 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
        ' Propagate 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 = mdsStepMaster
    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 = mdbaStepMaster.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) & _
'           " ) "
'
mdbaStepMaster.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 = mdbaStepMaster.CreateQueryDef(gstrEmptyString, strDelete)

' Call a procedure to assign the parameter values
Call AssignParameters(qy)

qy.Execute dbFailOnError
qy.Close

'
mdbaStepMaster.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 = mdbStepMaster.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)
    '
    mdbStepMaster.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 mdbStepMaster = 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 = "¥SMDData" & gsDefDBFileExt

Private Const merrFileNotFound As Integer = 3024
Private Const merrDaoTableMissing As Integer = 3078

Private Const STEPMASrTER_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_TSOL_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_DTL As String = "run_step_details"
Public Const TBL_CONNECTION_DTL 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 gcConnDtIs = New cConnDtIs
  Set gcConnDtIs.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, tsq_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, tsq_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
                Exit Function
            End If
    End Select

    UpgradeWsp.CommitTrans
    Exit Function

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

```

```

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
    End Sub

```

```

        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)
    End Select
CallFileDialogErr:
End Function

```

```

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
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 qryIt 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 qryIt = 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

```

```

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
    gintConnDtllLabel
    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
    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
    errDupConnDt1Name
    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  
errUnableToProcessListViewDbClick = 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

```

### SMRunOnly.vbw

```

Startup = 0, 0, 0, 0, C
Public = 0, 0, 0, 0, C
cAsyncShell = 0, 0, 0, 0, C
cFailedStep = 0, 0, 0, 0, C
cFailedSteps = 0, 0, 0, 0, C
cInstance = 0, 0, 0, 0, C
cStep = 0, 0, 0, 0, C
cArrConstraints = 0, 0, 0, 0, C
cArrParameters = 0, 0, 0, 0, C

```

```

cArrSteps = 0, 0, 0, 0, C
cNodeCollections = 0, 0, 0, 0, C
cParameter = 0, 0, 0, 0, C
cSequence = 0, 0, 0, 0, C
cStringSM = 0, 0, 0, 0, C
cIterator = 0, 0, 0, 0, C
cNode = 0, 0, 0, 0, C
cConstraint = 0, 0, 0, 0, C
cRunColIt = 0, 0, 0, 0, C
cStack = 0, 0, 0, 0, C
cVector = 0, 0, 0, 0, C
cRunItNode = 0, 0, 0, 0, C
cSubSteps = 0, 0, 0, 0, C
cSubStep = 0, 0, 0, 0, C
cRunItDetails = 0, 0, 0, 0, C
cFileSM = 0, 0, 0, 0, C
cVErrorsSM = 0, 0, 0, 0, C
SMErr = 0, 0, 0, 0, C
cTimerSM = 0, 0, 0, 0, C
DatabaseSM = 0, 0, 0, 0, C
DebugSM = 0, 0, 0, 0, C
MsgConfirm = 0, 0, 0, 0, C
cWorkspace = 0, 0, 0, 0, C
OpenFiles = 0, 0, 0, 0, C
cFileInfo = 0, 0, 0, 0, C
cVectorStr = 0, 0, 0, 0, C
SortSM = 0, 0, 0, 0, C
cRunWorkspace = 0, 0, 0, 0, C
cRunInst = 0, 0, 0, 0, C
cStepTree = 0, 0, 0, 0, C
cInstances = 0, 0, 0, 0, C
cVectorLng = 0, 0, 0, 0, C
cRunStep = 0, 0, 0, 0, C
cManager = 0, 0, 0, 0, C
cWorker = 0, 0, 0, 0, C
cGlobalStep = 0, 0, 0, 0, C
cRunOnly = 0, 0, 0, 0, C
ShellSM = 0, 0, 0, 0, C
TimerSM = 0, 0, 0, 0, C
frmSplash = 0, 0, 0, 0, C, 132, 174, 863, 764, C
WindowsApiCommon = 0, 0, 0, 0, C
StepCommon = 0, 0, 0, 0, C
FileCommon = 0, 0, 0, 0, C
WorkspaceCommon = 0, 0, 0, 0, C
IteratorCommon = 0, 0, 0, 0, C
ParameterCommon = 0, 0, 0, 0, C
Common = 0, 0, 0, 0, C
cTermSteps = 0, 0, 0, 0, C
cTermProcess = 0, 0, 0, 0, C
cTermStep = 0, 0, 0, 0, C
RunInstHelper = 0, 0, 0, 0, C
cConnections = 0, 0, 0, 0, C
cConnection = 0, 0, 0, 0, C

```

```

ToolsCommon = 0, 0, 0, 0, C
ConnectionCommon = 0, 0, 0, 0, C
cConnDtl = 0, 0, 0, 0, C
cConnDtIs = 0, 0, 0, 0, C
ConnDtlCommon = 0, 0, 0, 0, C
RunCommon = 0, 0, 0, 0, C
frmWorkspaceOpen = 0, 0, 0, 0, C, 176, 232, 907, 822, C

```

### 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
#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)
        End If
    End If
End If

End Function

```

```

        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
End Function

```

Exit Function

LoadRecordsetInStepsArrayErr:

```
LogErrors Errors
gstrSource = mstrModuleName & "LoadRecordsetInStepsArray"
On Error GoTo 0
Err.Raise vbObjectError + errLoadRsInArrayFailed, gstrSource, _
    LoadResString(errLoadRsInArrayFailed)
```

End Function

### TimerSM.bas

```
Attribute VB_Name = "TimerSM"
' FILE:      TimerSM.bas
'           Microsoft TPC-H Kit Ver. 1.00
'           Copyright Microsoft, 1999
'           All Rights Reserved
'
'
' PURPOSE:   This module contains wrapper functions for Timer APIs.
' Contact:   Reshma Tharamal (reshmat@microsoft.com)
```

Option Explicit

```
Private Declare Function SetTimer Lib "user32" (ByVal hWnd As Long, _
    ByVal nIDEvent As Long, ByVal uElapse As Long, ByVal lpTimerFunc As Long) _
    As Long
Private Declare Function KillTimer Lib "user32" (ByVal hWnd As Long, _
    ByVal nIDEvent As Long) As Long
Private Declare Sub CopyMemory Lib "kernel32" Alias "RtlMoveMemory" ( _
    pDest As Any, pSource As Any, ByVal ByteLen As Long)
```

Public gcTimerObjects As Collection

```
Private Sub TimerProc(ByVal lHwnd As Long, ByVal lMsg As Long, _
    ByVal lTimerID As Long, ByVal lTime As Long)
```

```
Dim nPtr As Long
Dim oTimerObject As cTimerSM
```

```
'Create a Timer object from the pointer
nPtr = gcTimerObjects.Item(Str$(lTimerID))
CopyMemory oTimerObject, nPtr, 4
'Call a method which will fire the Timer event
oTimerObject.Tick
'Get rid of the Timer object so that VB will not try to release it
CopyMemory oTimerObject, 0&, 4
```

End Sub

```
Public Function StartTimer(lInterval As Long) As Long
```

```
StartTimer = SetTimer(0, 0, lInterval, AddressOf TimerProc)
End Function
```

```
Public Sub StopTimer(lTimerID As Long)
KillTimer 0, lTimerID
End Sub
```

```
Public Sub SetInterval(lInterval As Long, lTimerID As Long)
SetTimer 0, lTimerID, lInterval, AddressOf TimerProc
End Sub
```

### ToolsCommon.bas

```
Attribute VB_Name = "ToolsCommon"
' FILE:      ToolsCommon.bas
'           Microsoft TPC-H Kit Ver. 1.00
'           Copyright Microsoft, 1999
'           All Rights Reserved
'
'
' PURPOSE:   Contains functions to remove run history and initialize
'           table creation scripts
' Contact:   Reshma Tharamal (reshmat@microsoft.com)
```

Option Explicit

Public sCreateTables() As String

```
Public Const gsExtSeparator As String = "."
```

```
Public Sub DeleteRunHistory(dbFile As DAO.Database)
' Delete all run history records from the database, viz. the records in
' run_header, run_step_details and run_parameters
```

```
Dim sDelete As String
```

```
On Error GoTo DeleteRunHistoryErr
```

```
sDelete = "delete from run_header "
dbFile.Execute sDelete, dbFailOnError
```

```
sDelete = "delete from run_step_details "
dbFile.Execute sDelete, dbFailOnError
```

```
sDelete = "delete from run_parameters "
dbFile.Execute sDelete, dbFailOnError
```

```
sDelete = "update att_identifiers " & _
    " set run_id = " & CStr(glMinId)
dbFile.Execute sDelete, dbFailOnError
```

Exit Sub

```

DeleteRunHistoryErr:
    LogErrors Errors
    Err.Raise vbObjectError + errDeleteDBRecordFailed, "DeleteRunHistory", _
        LoadResString(errDeleteDBRecordFailed)
End Sub

Public Function CreateConnectionsTableScript() As String
    ' Returns the table creation script for the workspace_connections table

    Call InitCreateSQLArray
    CreateConnectionsTableScript = sCreateTables(10)
    ReDim sCreateTables(0)

End Function

Public Function CreateConnectionDtlsTableScript() As String
    ' Returns the table creation script for the connection_dtls table

    Call InitCreateSQLArray
    CreateConnectionDtlsTableScript = sCreateTables(11)
    ReDim sCreateTables(0)

End Function

Public Sub InitCreateSQLArray()

ReDim sCreateTables(0 To 11)

sCreateTables(0) = "Create table att_identifiers (" & _
    "workspace_id Long, " & _
    "parameter_id Long, " & _
    "step_id Long, " & _
    "constraint_id Long, " & _
    "run_id Long, " & _
    "connection_id Long " & _
    ", " & FLD_ID_CONN_NAME & gstrBlank & "DATA_TYPE_LONG & _
    ");"

sCreateTables(1) = "Create table att_steps (step_id Long, " & _
    "version_no Text(255), " & _
    "step_label Text(255), " & _
    "step_file_name Text(255), " & _
    "step_text Memo, " & _
    "start_directory Text(255), " & _
    "workspace_id Long, " & _
    "parent_step_id Long, " & _
    "parent_version_no Text(255), " & _
    "step_level Long, " & _
    "sequence_no Integer, " & _
    "enabled_flag Bit, " & _
    "degree_parallelism Text(255), " & _
    "execution_mechanism Text(50), " & _
    "failure_details Text(255), " & _

```

```

    "continuation_criteria Text(50), " & _
    "global_flag Long, " & _
    "archived_flag Bit, " & _
    "output_file_name Text(255), " & _
    "error_file_name Text(255), " & _
    "iterator_name Text(255), " & _
    "CONSTRAINT pk_steps PRIMARY KEY (step_id, version_no) " & _
    ");"

    "log_file_name Text(255), " & _

sCreateTables(2) = "Create table att_workspaces (" & _
    "workspace_id Long, " & _
    "workspace_name Text(255), " & _
    "archived_flag Bit, " & _
    "CONSTRAINT pk_workspaces PRIMARY KEY (workspace_id) " & _
    ");"

sCreateTables(3) = "Create table iterator_values (" & _
    "step_id Long, " & _
    "version_no Text(255), " & _
    "type Integer, " & _
    "iterator_value Text(255), " & _
    "sequence_no Integer " & _
    ");"

sCreateTables(4) = "Create table run_header (" & _
    "run_id Long, " & _
    "workspace_id Long, " & _
    "start_time Currency, " & _
    "end_time Currency, " & _
    "CONSTRAINT pk_run_header PRIMARY KEY (run_id) " & _
    ");"

sCreateTables(5) = "Create table run_parameters (" & _
    "run_id Long, " & _
    "parameter_name Text(255), " & _
    "parameter_value Text(255) " & _
    ");"

sCreateTables(6) = "Create table run_step_details (" & _
    "run_id Long, " & _
    "step_id Long, " & _
    "version_no Text(255), " & _
    "instance_id Long, " & _
    "parent_instance_id Long, " & _
    "command Memo, " & _
    "iterator_value Text(255), " & _
    "start_time Currency, " & _
    "end_time Currency, " & _
    "elapsed_time Long " & _
    ");"

```

```

sCreateTables(7) = "Create table step_constraints (" & _
    "constraint_id          Long, " & _
    "step_id                Long, " & _
    "version_no             Text(255), " & _
    "constraint_type        Integer, " & _
    "global_step_id         Long, " & _
    "global_version_no      Text(255), " & _
    "sequence_no            Integer " & _
");"

sCreateTables(8) = "Create table workspace_parameters (" & _
    "workspace_id           Long, " & _
    "parameter_id          Long, " & _
    "parameter_name         Text(255), " & _
    "parameter_value        Text(255), " & _
    "description            Text(255), " & _
    "parameter_type         Integer, " & _
    "CONSTRAINT pk_parameters PRIMARY KEY (parameter_id) " & _
");"

sCreateTables(9) = "Create table db_details (" & _
    "db_version             Text(50) " & _
");"

sCreateTables(10) = "Create table " & TBL_CONNECTION_STRINGS & " (" & _
    "workspace_id           Long, " & _
    "connection_id          Long, " & _
    "connection_name         Text(255), " & _
    "connection_value        Text(255), " & _
    "description            Text(255), " & _
    "no_count_display        Bit, " & _
    "no_execute              Bit, " & _
    "parse_query_only        Bit, " & _
    "ANSI_quoted_identifiers Bit, " & _
    "ANSI_nulls              Bit, " & _
    "show_query_plan         Bit, " & _
    "show_stats_time         Bit, " & _
    "show_stats_io           Bit, " & _
    "parse_odbc_msg_prefixes Bit, " & _
    "row_count               long, " & _
    "tsql_batch_separator    Text(255), " & _
    "query_time_out          long, " & _
    "server_language         Text(255), " & _
    "character_translation   Bit, " & _
    "regional_settings        Bit, " & _
    "CONSTRAINT pk_connections PRIMARY KEY (connection_id) " & _
");"

' 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.
' Specify a connection for each odbc step. If the connection is of type,
' static, it should be kept open till the step execution is complete.
sCreateTables(11) = "Create table " & TBL_CONNECTION_DTLS & " (" & _

```

```

    FLD_ID_WORKSPACE & gstrBlank & DATA_TYPE_LONG & ", " & _
    FLD_ID_CONN_NAME & gstrBlank & DATA_TYPE_LONG & ", " & _
    FLD_CONN_DTL_CONNECTION_NAME & gstrBlank & DATA_TYPE_TEXT255 & ", " & _
    FLD_CONN_DTL_CONNECTION_STRING & gstrBlank & DATA_TYPE_TEXT255 & ", " & _
    FLD_CONN_DTL_CONNECTION_TYPE & gstrBlank & DATA_TYPE_INTEGER & ", " & _
    "CONSTRAINT pk_connection_name PRIMARY KEY (" & FLD_ID_CONN_NAME & ")
" & _
");"

End Sub

```

### WindowsApiCommon.bas

```

Attribute VB_Name = "WindowsApiCommon"
' FILE:      WindowsApiCommon.bas
'           Microsoft TPC-H Kit Ver. 1.00
'           Copyright Microsoft, 1999
'           All Rights Reserved
'
' PURPOSE:   This module contains functions that are wrappers around the
'           Windows API and are used by both StepMaster and SMRunOnly.
' 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 = "WindowsApiCommon."

Public Type PROCESS_INFORMATION
    hProcess As Long
    hThread  As Long
    dwProcessID As Long
    dwThreadID As Long
End Type

' Used by GetShortName to return the short file name for a given file
Private Declare Function GetShortPathName Lib "kernel32" ( _
    Alias "GetShortPathNameA" (ByVal lpszLongPath As String, _
    ByVal lpszShortPath As String, ByVal cchBuffer As Long) As Long

Public Declare Function GetExitCodeProcess Lib "kernel32" ( _
    ByVal hProcess As Long, lpExitCode As Long) As Long
Public Declare Function TerminateProcess Lib "kernel32" ( _
    hProcess As Long, uExitCode As Long) As Long
Public Declare Function CloseHandle Lib "kernel32" ( _
    ByVal hObject As Long) As Long

Public Const NORMAL_PRIORITY_CLASS As Long = &H20&

```

```

Public Const INFINITE                As Long = -1&
Public Const STATUS_WAIT_0           As Long = &H0
Public Const STATUS_ABANDONED_WAIT_0 As Long = &H80
Public Const STATUS_USER_APC         As Long = &HC0
Public Const STATUS_TIMEOUT          As Long = &H102
Public Const STATUS_PENDING          As Long = &H103

Public Const WAIT_FAILED              As Long = &FFFFFFFF
Public Const WAIT_OBJECT_0            As Long = STATUS_WAIT_0
Public Const WAIT_TIMEOUT             As Long = STATUS_TIMEOUT

Public Const WAIT_ABANDONED           As Long = STATUS_ABANDONED_WAIT_0
Public Const WAIT_ABANDONED_0        As Long = STATUS_ABANDONED_WAIT_0

Public Const WAIT_IO_COMPLETION       As Long = STATUS_USER_APC
Public Const STILL_ACTIVE             As Long = STATUS_PENDING

Public Const PROCESS_QUERY_INFORMATION As Long = &H400
Public Const STANDARD_RIGHTS_REQUIRED As Long = &HF0000

'-----
-
'Declarations for shelling:

Public Type STARTUPINFO
    cb                As Long
    lpReserved        As String
    lpDesktop         As String
    lpTitle           As String
    dwX               As Long
    dwY               As Long
    dwXSize           As Long
    dwYSize           As Long
    dwXCountChars    As Long
    dwYCountChars    As Long
    dwFillAttribute  As Long
    dwFlags           As Long
    wShowWindow       As Integer
    cbReserved2       As Integer
    lpReserved2       As Long
    hStdInput         As Long
    hStdOutput        As Long
    hStdError         As Long
End Type

Public Declare Function WaitForSingleObject Lib "kernel32" ( _
    ByVal hProcess As Long, ByVal dwMilliseconds As Long) As Long

Public Declare Function InputIdle Lib "user32" Alias "WaitForInputIdle" ( _
    ByVal hProcess As Long, ByVal dwMilliseconds As Long) As Long

Public Declare Function CreateProcessA Lib "kernel32" ( _
    ByVal lpApplicationName As Long, ByVal lpCommandLine As String, _
    ByVal lpProcessAttributes As Long, ByVal lpThreadAttributes As Long, _
    ByVal bInheritHandles As Long, ByVal dwCreationFlags As Long, _
    ByVal lpEnvironment As Long, ByVal lpCurrentDirectory As Long, _
    lpStartupInfo As STARTUPINFO, lpProcessInformation As _
    PROCESS_INFORMATION) As Long

Public Declare Function GetLastError Lib "kernel32" () As Long

Private Type OPENFILENAME
    lStructSize As Long
    hwndOwner As Long
    hInstance As Long
    lpstrFilter As String
    lpstrCustomFilter As String
    nMaxCustFilter As Long
    nFilterIndex As Long
    lpstrFile As String
    nMaxFile As Long
    lpstrFileName As String
    nMaxFileName As Long
    lpstrInitialDir As String
    lpstrTitle As String
    Flags As Long
    nFileOffset As Integer
    nFileExtension As Integer
    lpstrDefExt As String
    lCustData As Long
    lpfnHook As Long
    lpTemplateName As Long
End Type

Private Declare Function GetOpenFileName Lib "COMDLG32" _
    Alias "GetOpenFileNameA" (file As OPENFILENAME) As Long

Private Declare Function lstrlen Lib "kernel32" (lpstr As String) As Long

Public Const MAX_PATH = 255

' Used when creating a process
Public Const SW_SHOWMINNOACTIVE = 7
Public Const STARTF_USESHOWWINDOW = &H1

Public Const MB_YESNOCANCEL = &H3&
Public Const MB_ABORTRETRYIGNORE = &H2&
Public Const MB_OK = &H0&

Public Const MB_APPLMODAL = &H0&

Public Const MB_ICONQUESTION = &H20&
Public Const MB_ICONEXCLAMATION = &H30&

Public Const IDABORT = 3

```



```

Public Const IDRETRY = 4
Public Const IDIGNORE = 5
Public Const IDYES = 6
Public Const IDNO = 7
Public Const IDCANCEL = 2

Private Declare Function MessageBox Lib "user32" Alias "MessageBoxA" ( _
    ByVal hWnd As Long, ByVal lpText As String, _
    ByVal lpCaption As String, ByVal wType As Long) As Long

Private Type SYSTEMTIME
    wYear As Integer
    wMonth As Integer
    wDayOfWeek As Integer
    wDay As Integer
    wHour As Integer
    wMinute As Integer
    wSecond As Integer
    wMilliseconds As Integer
End Type

Private Declare Function Get64BitTime Lib "smtime.dll" ( _
    ByVal lpInitTime As Any) As Currency

Public Function ShowMessageBox(hWnd As Long, strText As String, _
    strTitle As String, wType As Integer) As Long
    ' Using the Windows MessageBox Api since the VB MsgBox function suppresses
    ' all events
    ShowMessageBox = MessageBox(hWnd, ByVal strText, ByVal strTitle, wType)

    If ShowMessageBox = 0 Then
        LogSystemError
        Err.Raise vbObjectError + errConfirmFailed, App.EXENAME, _
            LoadResString(errConfirmFailed)
    End If
End Function

Public Function ShowFileDialog(ByVal strFilter As String, _
    ByVal strDialogTitle As String, ByVal lngFlags As Long, _
    Optional ByVal strOldFile As String = gstrEmptyString) As String
    ' Returns the file name selected by the user
    Dim strInitDir As String
    Dim intPos As Integer
    Dim opfile As OPENFILENAME
    Dim sFile As String

    On Error GoTo ShowFileDialogErr

    If Not StringEmpty(strOldFile) Then
        intPos = InstrR(strOldFile, gstrFileSeparator)
        If intPos > 0 Then
            strInitDir = Left$(strOldFile, intPos - 1)
        End If
    End If

    With opfile
        .lStructSize = Len(opfile)
        .Flags = lngFlags
        .lpstrInitialDir = strInitDir
        .lpstrTitle = strDialogTitle
        .lpstrFilter = MakeWindowsFilter(strFilter)
        sFile = strOldFile & String$(MAX_PATH - Len(strOldFile), 0)
        .lpstrFile = sFile
        .nMaxFile = MAX_PATH
    End With

    If GetOpenFileName(opfile) Then
        ShowFileDialog = Left$(opfile.lpstrFile, InStr(opfile.lpstrFile,
vbNullChar) - 1)
    Else
        ShowFileDialog = strOldFile
    End If

    Exit Function

ShowFileDialogErr:
    Call LogErrors(Errors)
    ' Reset the selection to the passed in file, if any
    ShowFileDialog = strOldFile
End Function

Private Function MakeWindowsFilter(sFilter As String) As String

    Dim s As String, ch As String, iTemp As Integer

    On Error GoTo MakeWindowsFilterErr

    ' To make Windows-style filter, replace | and : with nulls
    For iTemp = 1 To Len(sFilter)
        ch = Mid$(sFilter, iTemp, 1)
        If ch = "|" Then
            s = s & vbNullChar
        Else
            s = s & ch
        End If
    Next iTemp

    ' Put double null at end
    s = s & vbNullChar & vbNullChar
    MakeWindowsFilter = s

    Exit Function

MakeWindowsFilterErr:
    Call LogErrors(Errors)

```

```

gstrSource = mstrModuleName & "MakeWindowsFilter"
On Error GoTo 0
Err.Raise vbObjectError + errApiFailed, gstrSource, _
    LoadResString(errApiFailed)

End Function

Public Function GetShortName(ByVal sLongFileName As String) As String
' Returns the short name for the passed in file - will only work
' if the passed in path/file exists

Dim lRetVal As Long, sShortPathName As String, iLen As Integer
Dim sLongFile As String
Dim sDir As String
Dim sFile As String
Dim intPos As Integer

On Error GoTo GetShortNameErr

sFile = gstrEmptyString
sLongFile = MakePathValid(sLongFileName)
If StringEmpty(Dir$(sLongFile, vbNormal + vbDirectory)) Then
' The passed in path is a file that does not exist - since
' the GetShortPathName api does not work on non-existent files
' on Win2K, use the directory as an argument to the api and
' then append the file
intPos = InstrR(sLongFile, gstrFileSeparator)
sDir = Mid$(sLongFile, 1, intPos - 1)
sFile = Right(sLongFile, Len(sLongFile) - intPos + 1)
sLongFile = sDir
End If

'Set up buffer area for API function call return
sShortPathName = Space(MAX_PATH)
iLen = Len(sShortPathName)

'Call the function
lRetVal = GetShortPathName(sLongFile, sShortPathName, iLen)
If lRetVal = 0 Then
    Call LogSystemError
End If

GetShortName = IIf(lRetVal = 0, sLongFile, Left(sShortPathName, lRetVal))
If Not StringEmpty(sFile) Then
    GetShortName = GetShortName & sFile
End If

Exit Function

GetShortNameErr:
Call LogErrors(Errors)
gstrSource = mstrModuleName & "GetShortName"

```

```

On Error GoTo 0
Err.Raise vbObjectError + errApiFailed, gstrSource, _
    LoadResString(errApiFailed)

End Function
Public Function Determine64BitTime() As Currency

    Determine64BitTime = Get64BitTime(ByVal 0&)

End Function

```

### WorkspaceCommon.bas

```

Attribute VB_Name = "WorkspaceCommon"
' FILE: WorkspaceCommon.bas
' Microsoft TPC-H Kit Ver. 1.00
' Copyright Microsoft, 1999
' All Rights Reserved
'
'
' PURPOSE: Contains functionality common across StepMaster and
' SMRunOnly, pertaining to workspaces
' Specifically, functions to read workspace records from
' the database 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 = "WorkspaceCommon."

Public Function GetWorkspaceDetails(
Optional ByVal WorkspaceId As Long, _
Optional WorkspaceName As String = gstrEmptyString _
) As Variant
' Depending on the passed in parameter, it returns
' either the workspace name or the workspace identifier
' in a variant. The calling function must convert the
' return value to the appropriate type

Dim rstWorkspace As Recordset
Dim qyWsp As DAO.QueryDef
Dim strSQL As String
Dim cTempStr As cStringSM

On Error GoTo GetWorkspaceDetailsErr
gstrSource = mstrModuleName & "GetWorkspaceDetails"

If WorkspaceId = 0 And _
WorkspaceName = gstrEmptyString Then
    On Error GoTo 0

```

```

        Err.Raise vbObjectError + errMandatoryParameterMissing, _
            gstrSource, _
            LoadResString(errMandatoryParameterMissing)
    End If

    Set cTempStr = New cStringSM

    If WorkspaceId = 0 Then
        strSql = " Select workspace_id from att_workspaces " & _
            " where workspace_name = [w_name] "
        Set qyWsp = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)
        qyWsp.Parameters("w_name").Value = WorkspaceName
    Else
        strSql = " Select workspace_name from att_workspaces " & _
            " where workspace_id = [w_id] "
        Set qyWsp = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)
        qyWsp.Parameters("w_id").Value = WorkspaceId
    End If

    Set cTempStr = Nothing

    Set rstWorkspace = qyWsp.OpenRecordset(dbOpenForwardOnly)

    If rstWorkspace.RecordCount <> 0 Then
        GetWorkspaceDetails = rstWorkspace.Fields(0)
    Else
        rstWorkspace.Close
        qyWsp.Close
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidWorkspaceData, _
            gstrSource, _
            LoadResString(errInvalidWorkspaceData)
    End If

    rstWorkspace.Close
    qyWsp.Close
    Exit Function

GetWorkspaceDetailsErr:
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "GetWorkspaceDetails"
    On Error GoTo 0
    Err.Raise vbObjectError + errGetWorkspaceDetailsFailed, _
        gstrSource, _
        LoadResString(errGetWorkspaceDetailsFailed)

End Function

Public Sub ReadStepsInWorkspace(rstStepsInWorkSpace As Recordset, _
    qySteps As DAO.QueryDef, _
    Optional lngWorkspaceId As Long = glInvalidId, _
    Optional dbLoad As DAO.Database = Nothing, _
    Optional ByVal bSelectArchivedRecords As Boolean = False)

```

```

    ' This function will populate the passed in recordset with
    ' all the steps for a given workspace (if one is passed in, else all
    workspaces)

    Dim strSql As String

    On Error GoTo ReadStepsInWorkspaceErr

    ' Create a recordset object to retrieve all steps for
    ' the given workspace
    strSql = "Select step_id, step_label, step_file_name, step_text, " & _
        " start_directory, version_no, workspace_id, " & _
        " parent_step_id, parent_version_no, " & _
        " sequence_no, step_level, " & _
        " enabled_flag, degree_parallelism, " & _
        " execution_mechanism, " & _
        " failure_details, continuation_criteria, " & _
        " global_flag, archived_flag, " & _
        " output_file_name, " & _
        " error_file_name, iterator_name " & _
        " from att_steps a " & _
        " where "

        ' log_file_name,

    If lngWorkspaceId <> glInvalidId Then
        strSql = strSql & " workspace_id = [w_id] AND "
    End If

    If Not bSelectArchivedRecords Then
        strSql = strSql & " archived_flag = [archived] AND "
    End If

    ' Find the highest X-component of the version number
    strSql = strSql & " 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
    strSql = strSql & " 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 ) ) "

' Append the order clause as follows
' First, separate all global/non-global steps
' Order the worker and manager steps by step_level to
' ensure that the parent steps are populated before
' any sub-steps within it
' Further ordering by parent_step_id and sequence_no
' ensures that all the children within a parent are
' selected in the necessary order
strSql = strSql & " order by global_flag, step_level, " & _
    " parent_step_id, sequence_no "

If dbLoad Is Nothing Then Set dbLoad = dbsAttTool

' Create a temporary Querydef object
Set qySteps = dbLoad.CreateQueryDef(gstrEmptyString, strSql)

' Initialize the parameter values
If lngWorkspaceId <> glInvalidId Then
    qySteps.Parameters("w_id").Value = lngWorkspaceId
End If

If Not bSelectArchivedRecords Then
    qySteps.Parameters("archived").Value = False
End If

Set rstStepsInWorkSpace = qySteps.OpenRecordset(dbOpenSnapshot)

Exit Sub

ReadStepsInWorkspaceErr:

LogErrors Errors
gstrSource = mstrModuleName & "ReadStepsInWorkspace"
On Error GoTo 0
Err.Raise vbObjectError + errReadWorkspaceDataFailed, _
    gstrSource, _
    LoadResString(errReadWorkspaceDataFailed)

End Sub
Public Sub ReadWorkspaces(dbLoad As Database, rstWsp As Recordset, _
    qyWsp As DAO.QueryDef, _
    Optional ByVal bSelectArchivedRecords As Boolean = False)

' This function will populate the passed in recordset with all workspace
records

Dim strSql As String

On Error GoTo ReadWorkspacesErr

' Create a recordset object containing all the workspaces
' (that haven't been archived) in the database
strSql = " Select workspace_id, workspace_name, archived_flag " & _
    " from att_workspaces "

If Not bSelectArchivedRecords Then
    strSql = strSql & " where archived_flag = [archived]"
End If
strSql = strSql & " order by workspace_name"

Set qyWsp = dbLoad.CreateQueryDef(gstrEmptyString, strSql)
If Not bSelectArchivedRecords Then
    qyWsp.Parameters("archived").Value = False
End If

Set rstWsp = qyWsp.OpenRecordset(dbOpenForwardOnly)

Exit Sub

ReadWorkspacesErr:

LogErrors Errors
gstrSource = mstrModuleName & "ReadWorkspaces"
On Error GoTo 0
Err.Raise vbObjectError + errReadWorkspaceDataFailed, _
    gstrSource, _
    LoadResString(errReadWorkspaceDataFailed)

End Sub
Public Sub ShowWorkspacesInDb(dbLoad As Database)

Dim recWorkspaces As Recordset
Dim qryAllWsp As QueryDef

On Error GoTo ShowWorkspacesInDbErr

' Set the mousepointer to indicate Busy
Call ShowBusy

Load frmWorkspaceOpen

Call ReadWorkspaces(dbLoad, recWorkspaces, qryAllWsp)

frmWorkspaceOpen.lstWorkspaces.Clear

' Load all the workspaces into the listbox
If recWorkspaces.RecordCount <> 0 Then
    Do
        ' Add the workspace name to the list and store
        ' the corresponding workspace id as the ItemData
        ' property of the item.
        ' The workspace id will be used for all further

```

```

        ' processing of the workspace
        frmWorkspaceOpen.lstWorkspaces.AddItem
recWorkspaces![workspace_name]

frmWorkspaceOpen.lstWorkspaces.ItemData(frmWorkspaceOpen.lstWorkspaces.NewIndex
) = -
        recWorkspaces![workspace_id]
        recWorkspaces.MoveNext

        Loop Until recWorkspaces.EOF
End If
recWorkspaces.Close
qryAllWsp.Close

' Reset the mousepointer
ShowFree

#If RUN_ONLY Then
    frmWorkspaceOpen.Show vbModal
#Else
    frmWorkspaceOpen.Show vbModal, frmMain
#End If

Exit Sub

ShowWorkspacesInDbErr:
    LogErrors Errors
    Call ShowFree
    Err.Raise vbObjectError + errProgramError, mstrModuleName &
"ShowWorkspacesInDb", _
        LoadResString(errProgramError)

End Sub
Private Sub ReadWorkspaceParameters(lngWorkspaceId As Long, _
    rstWorkSpaceParameters As Recordset, _
    qyWspParams As DAO.QueryDef)

    ' Will populate the recordset with all the parameters for
    ' a given workspace

    Dim strSql As String

    On Error GoTo ReadWorkspaceParametersErr

    strSql = "Select parameter_id, parameter_name, " & _
        " parameter_value, workspace_id, parameter_type, description " & _
        " from workspace_parameters " & _
        " where workspace_id = [w_id] " & _
        " order by parameter_name, parameter_value "

    ' Create a temporary Querydef object and initialize
    ' it's parameter values
    Set qyWspParams = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)

```

```

        qyWspParams.Parameters("w_id").Value = lngWorkspaceId

    Set rstWorkSpaceParameters = qyWspParams.OpenRecordset(dbOpenSnapshot)

    Exit Sub

ReadWorkspaceParametersErr:

    LogErrors Errors
    gstrSource = mstrModuleName & "ReadWorkspaceParameters"
    On Error GoTo 0
    Err.Raise vbObjectError + errReadWorkspaceDataFailed, _
        gstrSource, _
        LoadResString(errReadWorkspaceDataFailed)

End Sub
Private Sub ReadConnections(lngWorkspaceId As Long, rstConns As Recordset, _
    qyConns As DAO.QueryDef)

    ' Will populate the recordset with all the parameters for
    ' a given workspace

    Dim strSql As String

    On Error GoTo ReadWorkspaceParametersErr

    strSql = "Select connection_id, " & _
        " connection_name, connection_value, workspace_id, 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, tsq_batch_separator,
query_time_out, " & _
        " server_language, character_translation, regional_settings " & _
        " from workspace_connections " & _
        " where workspace_id = [w_id] " & _
        " order by connection_name, connection_value "

    ' Create a temporary Querydef object and initialize
    ' it's parameter values
    Set qyConns = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)
    qyConns.Parameters("w_id").Value = lngWorkspaceId

    Set rstConns = qyConns.OpenRecordset(dbOpenSnapshot)

    Exit Sub

ReadWorkspaceParametersErr:

    LogErrors Errors
    On Error GoTo 0

```

```

    Err.Raise vbObjectError + errReadWorkspaceDataFailed, _
        mstrModuleName & "ReadConnections", _
LoadResString(errReadWorkspaceDataFailed)
End Sub
Private Sub ReadConnectionDtls(lngWorkspaceId As Long, rstConns As Recordset, _
    qyConns As DAO.QueryDef)

    ' Will populate the recordset with all the connection_dtls records for
    ' a given workspace

    Dim strSql As String

    On Error GoTo ReadWorkspaceParametersErr

    strSql = "Select " & FLD_ID_CONN_NAME & ", " & _
        FLD_CONN_DTL_CONNECTION_NAME & ", " & _
        FLD_CONN_DTL_CONNECTION_STRING & ", " & _
        FLD_ID_WORKSPACE & ", " & _
        FLD_CONN_DTL_CONNECTION_TYPE & _
        " from " & TBL_CONNECTION_DTLS & _
        " where " & FLD_ID_WORKSPACE & " = [w_id] " & _
        " order by " & FLD_CONN_DTL_CONNECTION_NAME

    ' Create a temporary Querydef object and initialize
    ' it's parameter values
    Set qyConns = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)
    qyConns.Parameters("w_id").Value = lngWorkspaceId

    Set rstConns = qyConns.OpenRecordset(dbOpenSnapshot)

    Exit Sub
ReadWorkspaceParametersErr:

    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errReadWorkspaceDataFailed, _
        mstrModuleName & "ReadConnectionDtls", _
LoadResString(errReadWorkspaceDataFailed)
End Sub

Public Sub ReadWorkspaceData(lngWorkspaceId As Long, _
    cStepsCol As cArrSteps, _
    cParamsCol As cArrParameters, _
    cConsCol As cArrConstraints, _
    cConns As cConnections, _
    cConnDetails As cConnDtls, _
    rstStepsInWsp As Recordset, _
    qyStepsInWsp As DAO.QueryDef, _
    rstParamsInWsp As Recordset, _
    qyParamsInWsp As DAO.QueryDef, _

```

```

    rstConns As Recordset, _
    qyConns As DAO.QueryDef, _
    rstConnDtls As Recordset, _
    qyConnDtls As DAO.QueryDef)
    ' Loads the passed in structures with all the data for
    ' the workspace. It also initializes the recordsets
    ' with the step and parameter records for the workspace.

    On Error GoTo ReadWorkspaceDataErr

    ShowBusy

    Call ReadStepsInWorkspace(rstStepsInWsp, qyStepsInWsp, lngWorkspaceId)

    ' Load all the steps in the array
    LoadRecordsetInStepsArray rstStepsInWsp, cStepsCol

    ' Initialize the steps with all the iterator
    ' records for each step
    Call LoadIteratorsForWsp(cStepsCol, lngWorkspaceId, rstStepsInWsp)

    ReadWorkspaceParameters lngWorkspaceId, rstParamsInWsp, qyParamsInWsp

    ' Load all the workspace parameters in the array
    LoadRecordsetInParameterArray rstParamsInWsp, cParamsCol

    ' Read and load connection strings
    ReadConnections lngWorkspaceId, rstConns, qyConns

    LoadRecordsetInConnectionArray rstConns, cConns

    ' Read and load connection information
    ReadConnectionDtls lngWorkspaceId, rstConnDtls, qyConnDtls

    LoadRSInConnDtlArray rstConnDtls, cConnDetails

    ' Finally, load the step constraints collection class with
    ' all the constraints for the steps in the workspace
    cConsCol.LoadConstraints lngWorkspaceId, rstStepsInWsp

    ShowFree
    Exit Sub
ReadWorkspaceDataErr:

    ' Log the error code raised by Visual Basic
    ShowFree
    Call LogErrors(Errors)
    On Error GoTo 0
    gstrSource = mstrModuleName & "ReadWorkspaceData"
    Err.Raise vbObjectError + errReadWorkspaceDataFailed, _
        gstrSource, _
        LoadResString(errReadWorkspaceDataFailed)

```

End Sub

## Att\_workspaces

*workspace\_id workspace\_name*

11 ACID - TPC-H Kit V2.00

*archived\_flag*

0

*workspace\_id  
parameter\_name  
parameter\_value*

11 *parameter\_id =*  
SERVER\_NAME  
EBS64.

139 *parameter\_type =*

0

*workspace\_id  
parameter\_name  
parameter\_value*

11 *parameter\_id =*  
NUM\_STREAMS  
7

138 *parameter\_type =*

0

## Workspace\_parameters

*workspace\_id* 11 *parameter\_id =* 210 *parameter\_type =* 0  
*parameter\_name* saPASSWORD  
*parameter\_value* hitachi

*workspace\_id* 11 *parameter\_id =* 144 *parameter\_type =* 0  
*parameter\_name* DUR\_RUN\_ID  
*parameter\_value*

*workspace\_id* 11 *parameter\_id =* 143 *parameter\_type =* 3  
*parameter\_name* OUTPUT\_DIR  
*parameter\_value* C:¥MSTPCH~2.200¥OUTPUT¥42

*workspace\_id* 11 *parameter\_id =* 142 *parameter\_type =* 3  
*parameter\_name* RUN\_ID  
*parameter\_value* 42

*workspace\_id* 11 *parameter\_id =* 141 *parameter\_type =* 0  
*parameter\_name* KIT\_DIR  
*parameter\_value* C:¥MSTPCH\_Beta.200

*workspace\_id* 11 *parameter\_id =* 140 *parameter\_type =* 0  
*parameter\_name* TOOLS\_DIR  
*parameter\_value* %KIT\_DIR%¥tools



<i>workspace_id</i>	11	<i>parameter_id =</i>	137	<i>parameter_type =</i>	0
<i>parameter_name</i>		DEFAULT_DIR			
<i>parameter_value</i>		%KIT_DIR%¥OUTPUT			
<i>workspace_id</i>	11	<i>parameter_id =</i>	136	<i>parameter_type =</i>	0
<i>parameter_name</i>		DBNAME			
<i>parameter_value</i>		tpch1g			
<i>workspace_id</i>	11	<i>parameter_id =</i>	135	<i>parameter_type =</i>	0
<i>parameter_name</i>		ACID_DIR			
<i>parameter_value</i>		%KIT_DIR%¥ACID			

## Connection\_dtls

<i>worksp</i>	<i>connection</i>	<i>connection_name</i>	<i>connecti</i>	<i>on_type</i>
<i>ace_id</i>	<i>_name_id</i>			
11	66	Static_Connection_to_DB_10	DBCONECTION	1
11	65	Static_Connection_to_DB_9	DBCONECTION	1
11	64	Static_Connection_to_DB_8	DBCONECTION	1
11	63	Static_Connection_to_DB_7	DBCONECTION	1
11	62	Static_Connection_to_DB_6	DBCONECTION	1
11	61	Static_Connection_to_DB_5	DBCONECTION	1
11	60	Static_Connection_to_DB_4	DBCONECTION	1
11	59	Static_Connection_to_DB_3	DBCONECTION	1
11	58	Static_Connection_to_DB_2	DBCONECTION	1
11	57	Static_Connection_to_DB_1	DBCONECTION	1
11	56	Dynamic_Connection_to_DB	DBCONECTION	2

## Workspace\_connections

*workspace\_id* 11  
*connection\_id* 17  
*connection\_name* DBCONNECTION  
*connection\_value* DRIVER=SQL  
*description*  
*no\_count\_display* 0  
*no\_execute* 0  
*parse\_query\_only* 0  
*ANSI\_quoted\_identifiers* 0  
*ANSI\_nulls* -1  
*show\_query\_plan* 0  
*show\_stats\_time* 0  
*show\_stats\_io* 0  
*parse\_odbc\_msg\_prefixes* -1  
*row\_count* 0  
*tsql\_batch\_separator* GO  
*query\_time\_out* 0  
*server\_language* (Default)  
*character\_translation* -1  
*regional\_settings* 0

*execution\_mechanism* = 0

*continuation\_criteria* = 0

*failure\_details* =

*step\_file\_name* =

*version\_no* =

6.0

*start\_directory* =

*parent\_version\_no* =

0.0

*step\_text* =

## 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*

= -1

*iterator\_name* =

*degree\_parallelism* 1

<i>step_label</i> = TPC-H Atomicity	<i>step_id</i> = 921	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 2	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= -1		<i>failure_details</i> =	
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>step_file_name</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>version_no</i> =	
<i>failure_details</i> =		3.14	
<i>step_file_name</i> =		<i>start_directory</i> =	
<i>version_no</i> =	3.0	<i>parent_version_no</i> =	0.0
<i>start_directory</i> =		<i>step_text</i> =	
<i>parent_version_no</i> =	0.0	<i>step_label</i> = TPC-H Durability (Pre-Test Components)	<i>step_id</i> = 924
<i>step_text</i> =		<i>global_flag</i> = 0	
		<i>sequence_no</i> = 5	<i>step_level</i> = 0
		= 0	<i>parent_step_id</i> = 0
		<i>enabled_flag</i>	
<i>step_label</i> = TPC-H Isolation	<i>step_id</i> = 922	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 3	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= -1		<i>failure_details</i> =	
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>step_file_name</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>version_no</i> =	6.0
<i>failure_details</i> =		<i>start_directory</i> =	
<i>step_file_name</i> =		<i>parent_version_no</i> =	0.0
<i>version_no</i> =	3.0	<i>step_text</i> =	
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = TPC-H Consistency	<i>step_id</i> = 923		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 4	<i>step_level</i> = 0		
= -1			

```

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
= 0
iterator_name = degree_parallelism 1
execution_mechanism = 0 continuation_criteria = 0
failure_details =
step_file_name =
version_no = 8.0
start_directory =
parent_version_no = 0.0
step_text =

iterator_name = degree_parallelism 1
execution_mechanism = 1 continuation_criteria = 2
failure_details =
step_file_name = %ACID_DIR%\%acidproc%\%acid_hist.sql
version_no = 1.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 6.0
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 = 1.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 6.0
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 = 3.0
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

```

*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* = 2.0  
*start\_directory* = %ACID\_DIR%  
*parent\_version\_no* = 6.0  
*step\_text* = sqlcmd -Usa -P%saPASSWORD% -S%SERVER\_NAME% -d%DBNAME% -  
 iacidproc¥%PROC\_NAME%

*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* = 3.0  
*step\_text* =

*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* = 6.0  
*step\_text* = sqlcmd -Usa -P%saPASSWORD% -S%SERVER\_NAME% -d%DBNAME% -w 128 -  
 Q"tpch\_acid\_values  
 %NUM\_STREAMS%"

```

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 = 3.0
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 = 3.0
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 = 3.0
step_text = %TOOLS_DIR%\Utility\kill_semaphore.exe

```

<i>step_label</i> = Isolation 1	<i>step_id</i> = 935	<i>iterator_name</i> =	<i>degree_parallelism</i> = 3
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 2	<i>step_level</i> = 1	<i>parent_step_id</i> = 922	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 3	<i>start_directory</i> =	<i>parent_version_no</i> = 3.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>step_text</i> =	
<i>failure_details</i> =			
<i>step_file_name</i> =			
<i>version_no</i> =	0.0	<i>step_label</i> = Isolation 4	<i>step_id</i> = 938
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	3.0	<i>sequence_no</i> = 5	<i>step_level</i> = 1
<i>step_text</i> =		= -1	<i>parent_step_id</i> = 922
		<i>enabled_flag</i>	
		<i>iterator_name</i> =	<i>degree_parallelism</i> = 3
<i>step_label</i> = Isolation 2	<i>step_id</i> = 936	<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>global_flag</i> = 0		<i>failure_details</i> =	
<i>sequence_no</i> = 3	<i>step_level</i> = 1	<i>parent_step_id</i> = 922	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 3	<i>start_directory</i> =	<i>parent_version_no</i> = 3.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>step_text</i> =	
<i>failure_details</i> =			
<i>step_file_name</i> =			
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	3.0		
<i>step_text</i> =			
<i>step_label</i> = Isolation 3	<i>step_id</i> = 937		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 4	<i>step_level</i> = 1	<i>parent_step_id</i> = 922	<i>enabled_flag</i>
= -1			

```

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 = %ACID_DIR%
parent_version_no = 3.14
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 = 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 = 3.0
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

```



```

--
TRUNCATE TABLE HISTORY_ACI

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 = 3.0
start_directory = %ACID_DIR%
parent_version_no = 3.14
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 = 3.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 3.14
step_text =
--
-- This worker will clear the HISTORY_ACI table used during the durability process

```

```

step_label = ACID Transaction Streams          step_id = 945
global_flag = 0
sequence_no = 6 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 = 2.0
start_directory = parent_version_no = 3.14
step_text =

step_label = Execute Consistency Test 2      step_id = 946
global_flag = 0
sequence_no = 7 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 = 2.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 3.14
step_text =

step_label = Post-Process Consistency Output  step_id = 947
global_flag = 0
sequence_no = 9 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 = 2.0
start_directory = parent_version_no = 3.14
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 = 1.0
start_directory = %ACID_DIR%
parent_version_no = 6.0
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 = 1.0
start_directory = %ACID_DIR%
parent_version_no = 6.0
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 = 4.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 6.0
step_text = --
-- This worker will clear the HISTORY_DUR table used during the durability process

--
TRUNCATE TABLE HISTORY_DUR
select convert(char(30), getdate(),9)
SELECT *
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 = 1.0
start_directory = Static_Connection_to_DB_1
parent_version_no = 6.0
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 = 1.0
start_directory =
parent_version_no = 6.0
step_text =

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 = 8.0
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

```

drop table acid\_values

*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* = 3.0  
*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* = 3.0  
*step\_text* = if exists ( select name from sysobjects where name = '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* = 3.0  
*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 = 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 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

```



```

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 =
execution_mechanism =
failure_details =
step_file_name =
version_no =
start_directory =
parent_version_no =
step_text =

iterator_name =
execution_mechanism = 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 =

```

<i>step_label</i> = Isolation 6_2	<i>step_id</i> = 975	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 3	<i>step_level</i> = 2	<i>parent_step_id</i> = 940	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 1.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	2.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =			
<i>version_no</i> =	0.0	<i>step_label</i> = Consistency Stream 2	<i>step_id</i> = 978
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.0	<i>sequence_no</i> = 2	<i>step_level</i> = 2
<i>step_text</i> =		= -1	<i>parent_step_id</i> = 945
		<i>enabled_flag</i>	
<i>step_label</i> = Isolation 6_3	<i>step_id</i> = 976	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 4	<i>step_level</i> = 2	<i>parent_step_id</i> = 940	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 0.1
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	2.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =			
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Consistency Stream 1	<i>step_id</i> = 977		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 1	<i>step_level</i> = 2	<i>parent_step_id</i> = 945	<i>enabled_flag</i>
= -1			

<i>step_label</i> = Consistency Stream 3	<i>step_id</i> = 979	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 3	<i>step_level</i> = 2	<i>parent_step_id</i> = 945	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	2.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0	<i>step_label</i> = Consistency Stream 6	<i>step_id</i> = 982
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	2.0	<i>sequence_no</i> = 6	<i>step_level</i> = 2
<i>step_text</i> =		= -1	<i>parent_step_id</i> = 945
		<i>enabled_flag</i>	
<i>step_label</i> = Consistency Stream 4	<i>step_id</i> = 980	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 4	<i>step_level</i> = 2	<i>parent_step_id</i> = 945	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	1.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	2.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	2.0		
<i>step_text</i> =			
<i>step_label</i> = Consistency Stream 5	<i>step_id</i> = 981		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 5	<i>step_level</i> = 2	<i>parent_step_id</i> = 945	<i>enabled_flag</i>
= -1			

<i>step_label</i> = Consistency Stream 7	<i>step_id</i> = 983	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 7	<i>step_level</i> = 2	<i>parent_step_id</i> = 945	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	2.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =		<i>step_label</i> = Consistency Stream 10	<i>step_id</i> = 986
<i>version_no</i> = 0.0		<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 10	<i>step_level</i> = 2
<i>parent_version_no</i> = 2.0		= 0	<i>parent_step_id</i> = 945
<i>step_text</i> =		<i>enabled_flag</i>	
<i>step_label</i> = Consistency Stream 8	<i>step_id</i> = 984	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 8	<i>step_level</i> = 2	<i>parent_step_id</i> = 945	<i>enabled_flag</i>
= 0		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	2.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =		<i>step_label</i> = Consistency Stream 9	<i>step_id</i> = 985
<i>version_no</i> = 0.0		<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 9	<i>step_level</i> = 2
<i>parent_version_no</i> = 2.0		= 0	<i>parent_step_id</i> = 945
<i>step_text</i> =		<i>enabled_flag</i>	

<i>step_label</i> = Durability Stream 1	<i>step_id</i> = 987	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 1	<i>step_level</i> = 2	<i>parent_step_id</i> = 952	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 1.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	<i>parent_version_no</i> = 1.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>step_text</i> =	
<i>failure_details</i> =			
<i>step_file_name</i> =		<i>step_label</i> = Durability Stream 4	<i>step_id</i> = 990
<i>version_no</i> = 1.0		<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 4	<i>step_level</i> = 2
<i>parent_version_no</i> = 1.0		= -1	<i>parent_step_id</i> = 952
<i>step_text</i> =		<i>enabled_flag</i>	
		<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>step_label</i> = Durability Stream 2	<i>step_id</i> = 988	<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>global_flag</i> = 0		<i>failure_details</i> =	
<i>sequence_no</i> = 2	<i>step_level</i> = 2	<i>step_file_name</i> =	<i>version_no</i> = 2.0
= -1		<i>parent_step_id</i> = 952	<i>enabled_flag</i>
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	<i>parent_version_no</i> = 1.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>step_text</i> =	
<i>failure_details</i> =			
<i>step_file_name</i> =			
<i>version_no</i> = 1.0			
<i>start_directory</i> =			
<i>parent_version_no</i> = 1.0			
<i>step_text</i> =			
<i>step_label</i> = Durability Stream 3	<i>step_id</i> = 989		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 3	<i>step_level</i> = 2	<i>parent_step_id</i> = 952	<i>enabled_flag</i>
= -1			

<i>step_label</i> = Durability Stream 5	<i>step_id</i> = 991	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 5	<i>step_level</i> = 2	<i>parent_step_id</i> = 952	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 2.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	1.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =		<i>step_label</i> = Durability Stream 8	<i>step_id</i> = 994
<i>version_no</i> = 2.0		<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 8	<i>step_level</i> = 2
<i>parent_version_no</i> = 1.0		= 0	<i>parent_step_id</i> = 952
<i>step_text</i> =		<i>enabled_flag</i>	
<i>step_label</i> = Durability Stream 6	<i>step_id</i> = 992	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 6	<i>step_level</i> = 2	<i>parent_step_id</i> = 952	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	1.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =		<i>step_label</i> = Durability Stream 7	<i>step_id</i> = 993
<i>version_no</i> = 2.0		<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 7	<i>step_level</i> = 2
<i>parent_version_no</i> = 1.0		= -1	<i>parent_step_id</i> = 952
<i>step_text</i> =		<i>enabled_flag</i>	

```

step_label = Durability Stream 9                step_id = 995
global_flag = 0
sequence_no = 9  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 = %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

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 = 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 = 1.0
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

```



```

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 = 0.3
start_directory =
parent_version_no = 0.0
step_text =

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 = 2.0
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 = 2.0
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 = 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
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
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
global_flag = 0
sequence_no = 1 step_level = 3 parent_step_id = 977 enabled_flag

```



*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* = 1.0  
*start\_directory* = Static\_Connection\_to\_DB\_2  
*parent\_version\_no* = 0.1  
*step\_text* = EXEC cons\_tran %COUNTER%

*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 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



*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%

*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* = 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

*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* = 1.0  
*start\_directory* = Static\_Connection\_to\_DB\_1  
*parent\_version\_no* = 1.0  
*step\_text* = EXEC dur\_tran %COUNTER%

*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* = 1.0  
*start\_directory* = Static\_Connection\_to\_DB\_3  
*parent\_version\_no* = 1.0  
*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* = 1.0  
*start\_directory* = Static\_Connection\_to\_DB\_2  
*parent\_version\_no* = 1.0  
*step\_text* = EXEC dur\_tran %COUNTER%

*step\_label* = Dur - ACID Transactions Stream 3 *step\_id* = 1039  
*global\_flag* = 0



*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* = 1.0  
*start\_directory* = Static\_Connection\_to\_DB\_7  
*parent\_version\_no* = 2.0  
*step\_text* = EXEC dur\_tran %COUNTER%

*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 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

```

=
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 = 0.3
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 = 0.3
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 = 0.3
step_text =

```

<i>step_label</i> = Dur - Stream 4 Output	<i>step_id</i> = 1050	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 4	<i>step_level</i> = 3	<i>parent_step_id</i> = 998	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	1.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	
<i>failure_details</i> =		<i>parent_version_no</i> =	0.3
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =		<i>step_label</i> = Dur - Stream 7 Output	<i>step_id</i> = 1053
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.3	<i>sequence_no</i> = 7	<i>step_level</i> = 3
<i>step_text</i> =		= -1	<i>parent_step_id</i> = 998
		<i>enabled_flag</i>	
<i>step_label</i> = Dur - Stream 5 Output	<i>step_id</i> = 1051	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 5	<i>step_level</i> = 3	<i>parent_step_id</i> = 998	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	1.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	
<i>failure_details</i> =		<i>parent_version_no</i> =	0.3
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =		<i>step_label</i> = Dur - Stream 6 Output	<i>step_id</i> = 1052
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.3	<i>sequence_no</i> = 6	<i>step_level</i> = 3
<i>step_text</i> =		= -1	<i>parent_step_id</i> = 998
		<i>enabled_flag</i>	



<i>step_label</i> = Dur - Stream 8 Output	<i>step_id</i> = 1054	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 8	<i>step_level</i> = 3	<i>parent_step_id</i> = 998	<i>enabled_flag</i>
= 0		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	<i>parent_version_no</i> = 0.3
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>step_text</i> =	
<i>failure_details</i> =			
<i>step_file_name</i> =		<i>step_label</i> = Cons - Stream 1 Output	<i>step_id</i> = 1057
<i>version_no</i> = 0.0		<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 1	<i>step_level</i> = 3
<i>parent_version_no</i> = 0.3		= -1	<i>parent_step_id</i> = 1000
<i>step_text</i> =		<i>enabled_flag</i>	
		<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>step_label</i> = Dur - Stream 9 Output	<i>step_id</i> = 1055	<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>global_flag</i> = 0		<i>failure_details</i> =	
<i>sequence_no</i> = 9	<i>step_level</i> = 3	<i>parent_step_id</i> = 998	<i>enabled_flag</i>
= 0		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	<i>parent_version_no</i> = 0.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>step_text</i> =	
<i>failure_details</i> =			
<i>step_file_name</i> =			
<i>version_no</i> = 0.0			
<i>start_directory</i> =			
<i>parent_version_no</i> = 0.3			
<i>step_text</i> =			
<i>step_label</i> = Dur - Stream 10 Output	<i>step_id</i> = 1056		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 10	<i>step_level</i> = 3	<i>parent_step_id</i> = 998	<i>enabled_flag</i>
= 0			

<i>step_label</i> = Cons - Stream 2 Output	<i>step_id</i> = 1058	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 2	<i>step_level</i> = 3	<i>parent_step_id</i> = 1000	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0	<i>step_label</i> = Cons - Stream 5 Output	<i>step_id</i> = 1061
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.0	<i>sequence_no</i> = 5	<i>step_level</i> = 3
<i>step_text</i> =		= -1	<i>parent_step_id</i> = 1000
		<i>enabled_flag</i>	
<i>step_label</i> = Cons - Stream 3 Output	<i>step_id</i> = 1059	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 3	<i>step_level</i> = 3	<i>parent_step_id</i> = 1000	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Cons - Stream 4 Output	<i>step_id</i> = 1060		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 4	<i>step_level</i> = 3	<i>parent_step_id</i> = 1000	<i>enabled_flag</i>
= -1			

<i>step_label</i> = Cons - Stream 6 Output	<i>step_id</i> = 1062	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 6	<i>step_level</i> = 3	<i>parent_step_id</i> = 1000	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	0.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =		<i>step_label</i> = Cons - Stream 9 Output	<i>step_id</i> = 1065
<i>version_no</i> =	0.0	<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 9	<i>step_level</i> = 3
<i>parent_version_no</i> =	0.0	= 0	<i>parent_step_id</i> = 1000
<i>step_text</i> =			<i>enabled_flag</i>
<i>step_label</i> = Cons - Stream 7 Output	<i>step_id</i> = 1063	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 7	<i>step_level</i> = 3	<i>parent_step_id</i> = 1000	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	0.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =		<i>step_label</i> = Cons - Stream 8 Output	<i>step_id</i> = 1064
<i>version_no</i> =	0.0	<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 8	<i>step_level</i> = 3
<i>parent_version_no</i> =	0.0	= 0	<i>parent_step_id</i> = 1000
<i>step_text</i> =			<i>enabled_flag</i>

*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* =

*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 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

*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 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 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 6 Output Files          step_id = 1072
global_flag = 0
sequence_no = 1 step_level = 4 parent_step_id = 1062 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¥Stream6.out TYPE %OUTPUT_DIR%¥Cons_-
            _ACID_Transactions_Stream_6.%COUNTER%.out >> CONS¥Stream6.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 7 Output Files          step_id = 1073
global_flag = 0
sequence_no = 1 step_level = 4 parent_step_id = 1063 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¥Stream7.out TYPE %OUTPUT_DIR%¥Cons_-
            _ACID_Transactions_Stream_7.%COUNTER%.out >> CONS¥Stream7.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* = 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 TYPE %DEFAULT\_DIR%¥%DUR\_RUN\_ID%¥Dur\_-  
 \_ACID\_Transactions\_Stream\_1.%COUNTER%.out >> DUR¥Stream1.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 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 TYPE %DEFAULT_DIR%¥%DUR_RUN_ID%¥Dur-
             _ACID_Transactions_Stream_2.%COUNTER%.out >> DUR¥Stream2.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 = 1.0
step_text = IF EXIST DUR¥Stream4.out TYPE %DEFAULT_DIR%¥%DUR_RUN_ID%¥Dur-
             _ACID_Transactions_Stream_4.%COUNTER%.out >> DUR¥Stream4.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 TYPE %DEFAULT_DIR%¥%DUR_RUN_ID%¥Dur-
             _ACID_Transactions_Stream_3.%COUNTER%.out >> DUR¥Stream3.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 = 1.0
step_text = IF EXIST DUR¥Stream5.out 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 = 1.0
step_text = IF EXIST DUR¥Stream6.out 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 = 1.0
step_text = IF EXIST DUR¥Stream7.out 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 TYPE %DEFAULT\_DIR%¥%DUR\_RUN\_ID%¥Dur-  
 \_ACID\_Transactions\_Stream\_8.%COUNTER%.out >> DUR¥Stream8.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 TYPE %DEFAULT\_DIR%¥%DUR\_RUN\_ID%¥Dur-  
 \_ACID\_Transactions\_Stream\_10.%COUNTER%.out >> DUR¥Stream10.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 TYPE %DEFAULT\_DIR%¥%DUR\_RUN\_ID%¥Dur-  
 \_ACID\_Transactions\_Stream\_9.%COUNTER%.out >> DUR¥Stream9.out

CHECKPOINT  
GO

*step\_label* = Execute Checkpoint2 *step\_id* = 1450  
*global\_flag* = 0  
*sequence\_no* = 8 *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* =  
13.0  
*start\_directory* = Dynamic\_Connection\_to\_DB  
*parent\_version\_no* = 3.14  
*step\_text* = use %DBNAME%  
GO  
checkpoint  
GO

## Iterator\_values

<i>workspace</i> <i>_id</i>	<i>step_id</i>	<i>version</i> <i>_no</i>	<i>type</i>	<i>iterator_value</i>
11	1069	0.0	2	200
11	1072	0.0	1	251
11	1071	0.0	3	1
11	1071	0.0	2	300

*step\_label* = Execute Checkpoint1 *step\_id* = 1451  
*global\_flag* = 0  
*sequence\_no* = 5 *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* =  
14.0  
*start\_directory* = Dynamic\_Connection\_to\_DB  
*parent\_version\_no* = 3.14  
*step\_text* = use %DBNAME%  
GO

11	1071	0.0	1 201	011	1074	0.0	1 351
11	1070	0.0	2 250	011	1074	0.0	2 450
11	1070	0.0	1 151	011	1074	0.0	3 1
11	1027	0.0	1 11	011	1073	0.0	3 1
11	1069	0.0	1 101	011	1073	0.0	1 301
11	1073	0.0	2 400	011	1046	0.0	2 2010
11	1069	0.0	3 1	011	1076	0.0	2 550
11	1068	0.0	3 1	0			
11	1068	0.0	2 150	0			
11	1068	0.0	1 51	0			
11	1067	0.0	1 11	0			
11	1067	0.0	2 110	0			
11	1070	0.0	3 1	0			
11	1075	0.0	2 500	0			
11	1078	0.0	3 1	0			
11	1077	0.0	1 11	0			
11	1077	0.0	2 210	0			
11	1077	0.0	3 1	0			
11	1076	0.0	1 451	0			
11	1076	0.0	3 1	0			
11	1072	0.0	2 350	0			
11	1075	0.0	3 1	0			
11	1072	0.0	3 1	0			
11	1075	0.0	1 401	0			

11	1030	0.0	1 151	011	1044	0.0	1 1411
11	1067	0.0	3 1	011	1033	0.0	1 301
11	1033	0.0	3 1	011	1044	0.0	3 1
11	1032	0.0	1 251	011	1046	0.0	3 1
11	1032	0.0	2 350	011	1036	0.0	2 550
11	1032	0.0	3 1	011	1036	0.0	3 1
11	1031	0.0	1 201	011	1035	0.0	1 401
11	1033	0.0	2 400	0			
11	1031	0.0	3 1	0			
11	1034	0.0	3 1	0			
11	1030	0.0	2 250	0			
11	1030	0.0	3 1	0			
11	1029	0.0	2 200	0			
11	1029	0.0	3 1	0			
11	1029	0.0	1 101	0			
11	1027	0.0	2 110	0			
11	1027	0.0	3 1	0			
11	1031	0.0	2 300	0			
11	1036	0.0	1 451	0			
11	1079	0.0	1 411	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	1035	0.0	3 1	011	941	2.0	3 1
11	1035	0.0	2 500	011	941	2.0	2 10
11	1034	0.0	1 351	011	942	3.0	1 1
11	1034	0.0	2 450	011	942	3.0	2 %NUM_STREAMS%
11	1044	0.0	2 1610	011	942	3.0	3 1
11	1043	1.0	1 1211	011	1037	1.0	2 210
11	949	1.0	3 1	011	1040	1.0	3 1
11	1040	1.0	2 810	0			
11	1041	1.0	3 1	0			
11	1041	1.0	2 1010	0			
11	1041	1.0	1 811	0			
11	1042	1.0	2 1210	0			
11	1040	1.0	1 611	0			
11	1042	1.0	3 1	0			
11	1039	1.0	1 411	0			
11	1043	1.0	3 1	0			
11	1043	1.0	2 1410	0			
11	948	1.0	3 1	0			
11	948	1.0	2 10	0			
11	948	1.0	1 1	0			
11	949	1.0	1 1	0			
11	1078	0.0	1 211	0			
11	1042	1.0	1 1011	0			
11	1038	1.0	2 410	0			

11	1037	1.0	1	11	011	1079	0.0	2	610
11	929	2.0	4	iso34_tran_w_com.sql	1011	1079	0.0	3	1
11	1038	1.0	3	1	011	941	2.0	1	1
11	1038	1.0	1	211	011	1082	0.0	1	1011
11	1028	1.0	1	51	011	929	2.0	4	acid_values.sql
11	1028	1.0	2	150	011	929	2.0	4	iso6_tran_w_com.sql
11	1028	1.0	3	1	011	929	2.0	4	dur_tran.sql
11	1039	1.0	3	1	0				
11	1039	1.0	2	610	0				
11	1037	1.0	3	1	0				
11	1081	0.0	2	1010	0				
11	949	1.0	2	%NUM_STREAMS%	0				
11	1083	0.0	2	1410	0				
11	1083	0.0	1	1211	0				
11	1083	0.0	3	1	0				
11	1082	0.0	2	1210	0				
11	1082	0.0	3	1	0				
11	1084	0.0	1	1411	0				
11	1081	0.0	1	811	0				
11	1084	0.0	2	1610	0				
11	1081	0.0	3	1	0				
11	1080	0.0	1	611	0				
11	1080	0.0	2	810	0				
11	1080	0.0	3	1	0				

11	929	2.0	4	cons_tran.sql	7
11	929	2.0	4	acid_dur_tran.sql	6
11	929	2.0	4	iso5_parts.sql	5
11	929	2.0	4	iso_query.sql	4
11	1084	0.0	3	1	0
11	929	2.0	4	tran_w_com.sql	2
11	1078	0.0	2	410	0
11	929	2.0	4	acid_tran.sql	1
11	1086	0.0	3	1	0
11	1086	0.0	2	2010	0
11	1086	0.0	1	1811	0
11	1085	0.0	3	1	0
11	1085	0.0	2	1810	0
11	1085	0.0	1	1611	0
11	929	2.0	4	tran_w_rb.sql	3



## *Workspace\_parameters*

<i>workspace_id</i>	14	<i>parameter_id =</i>	222	<i>parameter_type =</i>	0
<i>parameter_name</i>			RF_FLATFILE_DIR		
<i>parameter_value</i>			E:%mnt%ffs%14%RF_Flat_File_16		
<i>workspace_id</i>	14	<i>parameter_id =</i>	231	<i>parameter_type =</i>	0
<i>parameter_name</i>			FLATFILE_DIR_4		
<i>parameter_value</i>			E:%mnt%ffs%5%Flat_File_4		
<i>workspace_id</i>	14	<i>parameter_id =</i>	230	<i>parameter_type =</i>	3
<i>parameter_name</i>			RUN_ID		
<i>parameter_value</i>			6		
<i>workspace_id</i>	14	<i>parameter_id =</i>	229	<i>parameter_type =</i>	0
<i>parameter_name</i>			KIT_DIR		
<i>parameter_value</i>			C:%MSTPCH_Beta.200_load		
<i>workspace_id</i>	14	<i>parameter_id =</i>	228	<i>parameter_type =</i>	0
<i>parameter_name</i>			RUN_DIR		
<i>parameter_value</i>			%KIT_DIR%run		
<i>workspace_id</i>	14	<i>parameter_id =</i>	227	<i>parameter_type =</i>	0
<i>parameter_name</i>			QUERY_DIR		
<i>parameter_value</i>			%RUN_DIR%Queries		
<i>workspace_id</i>	14	<i>parameter_id =</i>	226	<i>parameter_type =</i>	0
<i>parameter_name</i>			TEMPLATE_DIR		
<i>parameter_value</i>			%RUN_DIR%templates		
<i>workspace_id</i>	14	<i>parameter_id =</i>	225	<i>parameter_type =</i>	0
<i>parameter_name</i>			MAX_STREAMS		

*workspace\_id*  
*parameter\_name*  
*parameter\_value*

14 *parameter\_id =*  
TRACEFLAGS  
-c -x -E

213 *parameter\_type =*

0

<i>workspace_id</i>	14	<i>parameter_id =</i>	223	<i>parameter_type =</i>	3	<i>parameter_value</i>	%KIT_DIR%¥Setup
<i>parameter_name</i>		DEFAULT_DIR				<i>workspace_id</i>	14
<i>parameter_value</i>		%KIT_DIR%¥OUTPUT				<i>parameter_id =</i>	214
						<i>parameter_name</i>	
						<i>parameter_value</i>	
<i>workspace_id</i>	14	<i>parameter_id =</i>	234	<i>parameter_type =</i>	0		
<i>parameter_name</i>		FLATFILE_DIR_1					
<i>parameter_value</i>		E:¥mnt¥ffs¥0¥Flat_File_1					
<i>workspace_id</i>	14	<i>parameter_id =</i>	221	<i>parameter_type =</i>	0		
<i>parameter_name</i>		TABLE_LOAD_PARALLELISM					
<i>parameter_value</i>		8					
<i>workspace_id</i>	14	<i>parameter_id =</i>	220	<i>parameter_type =</i>	0		
<i>parameter_name</i>		INDEX_CREATE_PARALLELISM					
<i>parameter_value</i>		8					
<i>workspace_id</i>	14	<i>parameter_id =</i>	219	<i>parameter_type =</i>	0		
<i>parameter_name</i>		VALIDATION_DIR					
<i>parameter_value</i>		%SETUP_DIR%¥Validation					
<i>workspace_id</i>	14	<i>parameter_id =</i>	218	<i>parameter_type =</i>	0		
<i>parameter_name</i>		UPDATE_SETS					
<i>parameter_value</i>		14					
<i>workspace_id</i>	14	<i>parameter_id =</i>	217	<i>parameter_type =</i>	0		
<i>parameter_name</i>		SCALEFACTOR					
<i>parameter_value</i>		300					
<i>workspace_id</i>	14	<i>parameter_id =</i>	216	<i>parameter_type =</i>	3		
<i>parameter_name</i>		OUTPUT_DIR					
<i>parameter_value</i>		C:¥MSTPCH~1.200¥OUTPUT¥6					
<i>workspace_id</i>	14	<i>parameter_id =</i>	215	<i>parameter_type =</i>	0		
<i>parameter_name</i>		SETUP_DIR					

<i>workspace_id</i>	14	<i>parameter_id =</i>	224	<i>parameter_type =</i>	0	<i>parameter_value</i>	E:\mnt\ffs\7\Flat_File_8
<i>parameter_name</i>		TOOLS_DIR				<i>workspace_id</i>	14
<i>parameter_value</i>		%KIT_DIR%\tools				<i>parameter_id =</i>	245
						<i>parameter_name</i>	FLATFILE_DIR_9
						<i>parameter_value</i>	E:\mnt\ffs\0\Flat_File_9
<i>workspace_id</i>	14	<i>parameter_id =</i>	242	<i>parameter_type =</i>	0		
<i>parameter_name</i>		SYSTEM_ARCHITECTURE					
<i>parameter_value</i>		ia64					
<i>workspace_id</i>	14	<i>parameter_id =</i>	252	<i>parameter_type =</i>	0		
<i>parameter_name</i>		FLATFILE_DIR_12					
<i>parameter_value</i>		E:\mnt\ffs\5\Flat_File_12					
<i>workspace_id</i>	14	<i>parameter_id =</i>	251	<i>parameter_type =</i>	0		
<i>parameter_name</i>		FLATFILE_DIR_13					
<i>parameter_value</i>		E:\mnt\ffs\2\Flat_File_13					
<i>workspace_id</i>	14	<i>parameter_id =</i>	250	<i>parameter_type =</i>	0		
<i>parameter_name</i>		FLATFILE_DIR_14					
<i>parameter_value</i>		E:\mnt\ffs\6\Flat_File_14					
<i>workspace_id</i>	14	<i>parameter_id =</i>	249	<i>parameter_type =</i>	0		
<i>parameter_name</i>		FLATFILE_DIR_15					
<i>parameter_value</i>		E:\mnt\ffs\3\Flat_File_15					
<i>workspace_id</i>	14	<i>parameter_id =</i>	248	<i>parameter_type =</i>	0		
<i>parameter_name</i>		FLATFILE_DIR_16					
<i>parameter_value</i>		E:\mnt\ffs\7\Flat_File_16					
<i>workspace_id</i>	14	<i>parameter_id =</i>	247	<i>parameter_type =</i>	0		
<i>parameter_name</i>		FLATFILE_DIR_7					
<i>parameter_value</i>		E:\mnt\ffs\3\Flat_File_7					
<i>workspace_id</i>	14	<i>parameter_id =</i>	246	<i>parameter_type =</i>	0		
<i>parameter_name</i>		FLATFILE_DIR_8					

<i>workspace_id</i>	14	<i>parameter_id =</i>	232	<i>parameter_type =</i>	0	<i>parameter_value</i>	16				
<i>parameter_name</i>		FLATFILE_DIR_3				<i>workspace_id</i>	14	<i>parameter_id =</i>	235	<i>parameter_type =</i>	0
<i>parameter_value</i>		E:\mnt\ffs\1\Flat_File_3				<i>parameter_name</i>		SERVER			
						<i>parameter_value</i>		.			
<i>workspace_id</i>	14	<i>parameter_id =</i>	243	<i>parameter_type =</i>	0						
<i>parameter_name</i>		saPASSWORD									
<i>parameter_value</i>		hitachi									
<i>workspace_id</i>	14	<i>parameter_id =</i>	233	<i>parameter_type =</i>	0						
<i>parameter_name</i>		FLATFILE_DIR_2									
<i>parameter_value</i>		E:\mnt\ffs\4\Flat_File_2									
<i>workspace_id</i>	14	<i>parameter_id =</i>	241	<i>parameter_type =</i>	0						
<i>parameter_name</i>		BCP_ROWS									
<i>parameter_value</i>		10000									
<i>workspace_id</i>	14	<i>parameter_id =</i>	240	<i>parameter_type =</i>	0						
<i>parameter_name</i>		FLAT_FILE_PARALLELISM									
<i>parameter_value</i>		1									
<i>workspace_id</i>	14	<i>parameter_id =</i>	239	<i>parameter_type =</i>	0						
<i>parameter_name</i>		FLATFILE_DIR_6									
<i>parameter_value</i>		E:\mnt\ffs\6\Flat_File_6									
<i>workspace_id</i>	14	<i>parameter_id =</i>	238	<i>parameter_type =</i>	0						
<i>parameter_name</i>		FLATFILE_DIR_5									
<i>parameter_value</i>		E:\mnt\ffs\2\Flat_File_5									
<i>workspace_id</i>	14	<i>parameter_id =</i>	237	<i>parameter_type =</i>	0						
<i>parameter_name</i>		TOTAL_LOAD_FILES_PER_TABLE									
<i>parameter_value</i>		16									
<i>workspace_id</i>	14	<i>parameter_id =</i>	236	<i>parameter_type =</i>	0						
<i>parameter_name</i>		FILES_PER_UPDATE_SET									

*workspace\_id* 14 *parameter\_id* = 253 *parameter\_type* = 0  
*parameter\_name* FLATFILE\_DIR\_11  
*parameter\_value* E:¥mnt¥ffs¥1¥Flat\_File\_11

*workspace\_id* 14 *parameter\_id* = 244 *parameter\_type* = 0  
*parameter\_name* FLATFILE\_DIR\_10  
*parameter\_value* E:¥mnt¥ffs¥4¥Flat\_File\_10

## Connection\_dtls

<i>worksp</i>	<i>connection</i>	<i>connection_name</i>	<i>connecti</i>	<i>on_type</i>
<i>connection_string_name</i>	<i>ace_id</i>	<i>_name_id</i>		
14	99	Dynamic_connection_to_MASTER_Node4	2	
MASTERDBCONNECTION_NODE			4	
14	98	Dynamic_connection_to_MASTER_Node3	2	
MASTERDBCONNECTION_NODE			3	
14	97	Dynamic_connection_to_MASTER_Node2	2	
MASTERDBCONNECTION_NODE			2	
14	96	Dynamic_connection_to_MASTER_Node1	2	
MASTERDBCONNECTION_NODE			1	
14	95	Static_Connection_to_Master		
MASTERDBCONNECTION 1				
14	94	Static_Connection_to_DB	DBCONNECTION	1
14	93	Dynamic_Connection_to_Master		
MASTERDBCONNECTION 2				
14	92	Dynamic_Connection_to_DB	DBCONNECTION	2

## Workspace\_connections

<i>workspace_id</i>	14
<i>connection_id</i>	27
<i>connection_name</i>	MASTERDBCONNECTION
<i>connection_value</i>	DRIVER=SQL Native Client;SERVER=%SERVER%;UID=sa;PWD=%saPASSWORD%;
<i>description</i>	
<i>no_count_display</i>	0
<i>no_execute</i>	0
<i>parse_query_only</i>	0
<i>ANSI_quoted_identifiers</i>	0
<i>ANSI_nulls</i>	-1
<i>show_query_plan</i>	0
<i>show_stats_time</i>	0
<i>show_stats_io</i>	0
<i>parse_odbc_msg_prefixes</i>	-1
<i>row_count</i>	0
<i>tsql_batch_separator</i>	GO
<i>query_time_out</i>	1
<i>server_language</i>	(Default)
<i>character_translation</i>	-1
<i>regional_settings</i>	0
<i>workspace_id</i>	14
<i>connection_id</i>	26
<i>connection_name</i>	DBCONNECTION
<i>connection_value</i>	DRIVER=SQL Native
<i>description</i>	
<i>no_count_display</i>	0
<i>no_execute</i>	0
<i>parse_query_only</i>	0
<i>ANSI_quoted_identifiers</i>	0
<i>ANSI_nulls</i>	-1

<i>show_query_plan</i>	0
<i>show_stats_time</i>	0
<i>show_stats_io</i>	0
<i>parse_odbc_msg_prefixes</i>	-1
<i>row_count</i>	0
<i>tsql_batch_separator</i>	GO
<i>query_time_out</i>	0
<i>server_language</i>	(Default)

*character\_translation* -1  
*regional\_settings* 0

## Att\_steps

*workspace\_id* = 14

*step\_label* = SqlServer Startup *step\_id* = 1452  
*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 /D "C:¥Program Files¥Microsoft SQL Server¥MSSQL.1¥MSSQL¥Binn" sqlservr -c  
%TRACEFLAGS%  
%TOOLS\_DIR%¥Utility¥wait4sql 40000

*step\_label* = SqlServer Shutdown *step\_id* = 1453  
*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%saPASSWORD% -t10 -Q"shutdown"  
%TOOLS\_DIR%¥Utility¥wait4sql 10000

*step\_label* = Wait For SQL Server *step\_id* = 1454  
*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

<i>step_label</i> = Generate FlatFiles	<i>step_id</i> = 1455	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 1	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= 0		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0	<i>step_label</i> = Configure SqlServer	<i>step_id</i> = 1458
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.0	<i>sequence_no</i> = 4	<i>step_level</i> = 0
<i>step_text</i> =		= -1	<i>parent_step_id</i> = 0
		<i>enabled_flag</i>	
<i>step_label</i> = Start SqlServer	<i>step_id</i> = 1456	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 2	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Create database	<i>step_id</i> = 1457		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 3	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= -1			



<i>step_label</i> = Create and Load Tables	<i>step_id</i> = 1459	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 5	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0	<i>step_label</i> = Backup TPC-H Database (If not used for ACID)	<i>step_id</i> = 1462
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.0	<i>sequence_no</i> = 8	<i>step_level</i> = 0
<i>step_text</i> =		= 0	<i>parent_step_id</i> = 0
		<i>enabled_flag</i>	
<i>step_label</i> = Create Indexes and Statistics	<i>step_id</i> = 1460	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 6	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Prep Database, End of Load	<i>step_id</i> = 1461		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 7	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= -1			

<i>step_label</i> = Capture Database and Table Space Used (Do Not Use on	<i>step_id</i> = 1463	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 9	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= 0		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	<i>parent_version_no</i> = 0.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>step_text</i> =	
<i>failure_details</i> =			
<i>step_file_name</i> =			
<i>version_no</i> =	0.0	<i>step_label</i> = Generate Query Plans	<i>step_id</i> = 1466
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.0	<i>sequence_no</i> = 12	<i>step_level</i> = 0
<i>step_text</i> =		= 0	<i>parent_step_id</i> = 0
		<i>enabled_flag</i>	
		<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>step_label</i> = Generate Queries via QGen	<i>step_id</i> = 1464	<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>global_flag</i> = 0		<i>failure_details</i> =	
<i>sequence_no</i> = 10	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	<i>parent_version_no</i> = 0.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>step_text</i> =	
<i>failure_details</i> =			
<i>step_file_name</i> =			
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Validation (for SCALEFACTOR=1 Only)	<i>step_id</i> = 1465		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 11	<i>step_level</i> = 0	<i>parent_step_id</i> = 0	<i>enabled_flag</i>
= 0			

```

step_label = Generate FlatFiles in Parallel          step_id = 1467
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1455 enabled_flag
= -1
iterator_name = degree_parallelism
%TOTAL_LOAD_FILES_PER_TABLE%
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 = Generate Update Files          step_id = 1468
global_flag = 0
sequence_no = 2 step_level = 1 parent_step_id = 1455 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 = %RF_FLATFILE_DIR%
parent_version_no = 0.0
step_text = %TOOLS_DIR%\%DBGEN%\%SYSTEM_ARCHITECTURE%\%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 = 1469
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1456 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 =
parent_version_no = 0.0
step_text = start /D "C:\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL\Binn" sqlservr
%TRACEFLAGS%

```

```

step_label = Create TPC-H Database                                step_id = 1470
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1457 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 = 0.0
start_directory = Static_Connection_to_Master
parent_version_no = 0.0
step_text =

step_label = Set Database Options                                step_id = 1471
global_flag = 0
sequence_no = 2 step_level = 1 parent_step_id = 1457 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 = Static_Connection_to_Master
parent_version_no = 0.0
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 = Set sp_configure Options                            step_id = 1472
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1458 enabled_flag
= -1
iterator_name = degree_parallelism 1
execution_mechanism = 1 continuation_criteria = 2
failure_details =
step_file_name = %SETUP_DIR%\Utility\conf_tpch.sql
version_no = 0.0
start_directory = Dynamic_Connection_to_Master
parent_version_no = 0.0
step_text =

```

*step\_label* = Move TempDB *step\_id* = 1473  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 1 *parent\_step\_id* = 1458 *enabled\_flag*  
= 0  
*iterator\_name* = *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* = %SETUP\_DIR%\%%DBNAME%\tempdb\MoveTempDB.sql  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*step\_text* =

*step\_label* = ReSize TempDB *step\_id* = 1474  
*global\_flag* = 0  
*sequence\_no* = 3 *step\_level* = 1 *parent\_step\_id* = 1458 *enabled\_flag*  
= 0  
*iterator\_name* = *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* = %SETUP\_DIR%\%%DBNAME%\tempdb\ReSizeTempDB.sql  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*step\_text* =

```

step_label = Load Start Timestamp                                step_id = 1475
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1459 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_Master
parent_version_no = 0.0
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 = (Drop/)Create Tables                                step_id = 1476
global_flag = 0
sequence_no = 2 step_level = 1 parent_step_id = 1459 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 =

```

<i>step_label</i> = Parallel Partitioned Table Load	<i>step_id</i> = 1477	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 1	<i>continuation_criteria</i> = 1
<i>sequence_no</i> = 3	<i>step_level</i> = 1	<i>parent_step_id</i> = 1459	<i>enabled_flag</i>
= -1		<i>failure_details</i> =	
<i>iterator_name</i> = TABLE	<i>degree_parallelism</i>	<i>step_file_name</i> = %SETUP_DIR%¥%DBNAME%¥CreateIndexes_1.sql	
%TABLE_LOAD_PARALLELISM%		<i>version_no</i> = 0.0	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> = Dynamic_Connection_to_DB	
<i>failure_details</i> =		<i>parent_version_no</i> = 0.0	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0		
<i>start_directory</i> =		<i>step_label</i> = Create FK Constraints	<i>step_id</i> = 1480
<i>parent_version_no</i> =	0.0	<i>global_flag</i> = 0	
<i>step_text</i> =		<i>sequence_no</i> = 2	<i>step_level</i> = 1
		= -1	<i>parent_step_id</i> = 1460
			<i>enabled_flag</i>
		<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
		<i>execution_mechanism</i> = 1	<i>continuation_criteria</i> = 2
		<i>failure_details</i> =	
<i>step_label</i> = Parallel Load Simple Tables	<i>step_id</i> = 1478	<i>step_file_name</i> = %SETUP_DIR%¥%DBNAME%¥createFK.sql	
<i>global_flag</i> = 0		<i>version_no</i> = 0.0	
<i>sequence_no</i> = 4	<i>step_level</i> = 1	<i>start_directory</i> = Dynamic_Connection_to_DB	
= -1		<i>parent_version_no</i> = 0.0	
<i>iterator_name</i> =	<i>degree_parallelism</i>	<i>step_text</i> =	
%TABLE_LOAD_PARALLELISM%			
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0		
<i>failure_details</i> =			
<i>step_file_name</i> =			
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Create Indexes 1	<i>step_id</i> = 1479		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 1	<i>step_level</i> = 1	<i>parent_step_id</i> = 1460	<i>enabled_flag</i>

```

=
step_label = Create Indexes 2 step_id = 1481 iterator_name = degree_parallelism 1
global_flag = 0 execution_mechanism = 1 continuation_criteria = 2
sequence_no = 3 step_level = 1 parent_step_id = 1460 enabled_flag failure_details =
= -1 step_file_name = version_no = 0.0
iterator_name = degree_parallelism 1 start_directory = Dynamic_Connection_to_DB
execution_mechanism = 1 continuation_criteria = 1 parent_version_no = 0.0
failure_details = step_text = alter database %DBNAME% set date_correlation_optimization ON
step_file_name = %SETUP_DIR%\%DBNAME%\CreateIndexes_2.sql
version_no = 0.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Create Statistics step_id = 1482
global_flag = 0
sequence_no = 4 step_level = 1 parent_step_id = 1460 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 = sp_createstats

step_label = Set Correlation step_id = 1483
global_flag = 0
sequence_no = 5 step_level = 1 parent_step_id = 1460 enabled_flag

```



```

step_label = Reset DB_Option                step_id = 1484
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1461 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 = 0.0
step_text = sp_dboption %DBNAME%, 'trunc. log on chkpt.',false
GO

```

```

step_label = Install Refresh Function Stored Procedures step_id = 1485
global_flag = 0
sequence_no = 2 step_level = 1 parent_step_id = 1461 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 = 0.0
step_text =

```

```

step_label = Backup TPC-H Database (Only if using backup to satisfy step_id = 1486
global_flag = 0

```

```

sequence_no = 3 step_level = 1 parent_step_id = 1461 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* = Restore DB *step\_id* = 1487  
*global\_flag* = 0  
*sequence\_no* = 4 *step\_level* = 1 *parent\_step\_id* = 1461 *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* = %SETUP\_DIR%  
*parent\_version\_no* = 0.0  
*step\_text* = osql -Usa -P%saPASSWORD% -e -i %SETUP\_DIR%¥¥DBNAME¥¥RestoreDatabase.sql  
 -o %OUTPUT\_DIR¥¥Restore\_DB.out

*step\_label* = Reconfiguration\_parameters *step\_id* = 1488  
*global\_flag* = 0  
*sequence\_no* = 5 *step\_level* = 1 *parent\_step\_id* = 1461 *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* = exec sp\_configure 'priority boost' ,1  
 execute sp\_configure 'max server memory' ,126000  
 reconfigure with override

*step\_label* = End of Load *step\_id* = 1489  
*global\_flag* = 0  
*sequence\_no* = 6 *step\_level* = 1 *parent\_step\_id* = 1461 *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* = 0.0  
*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

```

-- Calculate seed per clause 2.1.3.3
--
SET @seed0 =
CONVERT(integer,100000000*DATEPART(MM,@Finish)+1000000*DATEPART(DD,@Fini
sh)+10000*DATEP
ART(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 = Verify TPC-H Load                step_id = 1490
global_flag = 0
sequence_no = 7 step_level = 1 parent_step_id = 1461 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 = 0.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Generate QGEN Seed                step_id = 1491
global_flag = 0
sequence_no = 8 step_level = 1 parent_step_id = 1461 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 = DECLARE @Finish datetime
DECLARE @seed0 integer

--
-- Get ending time of database load
--
SELECT @Finish=LoadFinish FROM TPCH_AUX_TABLE

--

```

```

=
step_label = Set Correlation1 step_id = 1492 iterator_name = degree_parallelism 1
global_flag = 0 execution_mechanism = 1 continuation_criteria = 2
sequence_no = 1 step_level = 1 parent_step_id = 1462 enabled_flag failure_details =
= -1 step_file_name = %SETUP_DIR%\%%DBNAME%\BackupDatabase.sql
version_no = 0.0
iterator_name = degree_parallelism 1 start_directory = Dynamic_Connection_to_Master
execution_mechanism = 1 continuation_criteria = 2 parent_version_no = 0.0
failure_details = step_text =
step_file_name = version_no = 0.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text = alter database %%DBNAME%% set date_correlation_optimization ON

```

```

step_label = Create Backup Device(s) step_id = 1493
global_flag = 0
sequence_no = 2 step_level = 1 parent_step_id = 1462 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 = 0.0
step_text =

```

```

step_label = Execute the backup step_id = 1494
global_flag = 0
sequence_no = 3 step_level = 1 parent_step_id = 1462 enabled_flag

```

```

step_label = Execute DBCC UPDATEUSAGE          step_id = 1495
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1463 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 = --
-- Correct any potential inaccuracies in the system tables before running sp_spaceused
--
dbcc updateusage (%DBNAME%)
GO
dbcc updateusage (tempdb)
GO

```

```

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 SpaceUsed Procedure      step_id = 1496
global_flag = 0
sequence_no = 2 step_level = 1 parent_step_id = 1463 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_label = Execute Table Row Counts          step_id = 1497
global_flag = 0

sequence_no = 3 step_level = 1 parent_step_id = 1463 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 = 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(*) from PARTSUPP
GO
print 'Count of CUSTOMER Table'
GO
select count(*) from CUSTOMER
GO
print 'Count of ORDERS Table'
GO
select count(*) from ORDERS
GO

print 'Count of LINEITEM Table'
GO
select count(*) from LINEITEM
GO

step_label = Execute Log File Spaceused          step_id = 1498
global_flag = 0

sequence_no = 4 step_level = 1 parent_step_id = 1463 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 = dbcc sqlperf(logspace)

```

```

step_label = Execute TempDB spaceused          step_id = 1499
global_flag = 0
sequence_no = 5 step_level = 1 parent_step_id = 1463 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_Master
parent_version_no = 0.0
step_text = use tempdb
GO
sp_spaceused
GO
use %DBNAME%
GO

```

```

step_text = -- Update the benchmark auxiliary table
-- Use the Compaq seed for the 1253.3 QphH result (load end at (1999) 11-01 18:01:42
--
UPDATE TPCH_AUX_TABLE SET QgenSeed=1101180142
SELECT * from TPCH_AUX_TABLE

```

```

step_label = Set SEED to Compaq/1253.3 seed    step_id = 1500
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1464 enabled_flag
= 0
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_label = Generate Power Queries          step_id = 1501
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1464 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* = Generate Stream Queries *step\_id* = 1502  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 1 *parent\_step\_id* = 1464 *enabled\_flag*  
 = -1  
*iterator\_name* = STREAM\_NUM *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* =

*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* = Generate Validation Queries via QGEN *step\_id* = 1503  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 1 *parent\_step\_id* = 1465 *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* = 0.0  
*step\_text* =

*step\_label* = Execute Validation Queries *step\_id* = 1504  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 1 *parent\_step\_id* = 1465 *enabled\_flag*  
 = -1



```

step_label = Relocate Validation Query Output Files          step_id = 1505
global_flag = 0
sequence_no = 3 step_level = 1 parent_step_id = 1465 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 = 0.0
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 = 1506
global_flag = 0
sequence_no = 1 step_level = 1 parent_step_id = 1466 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 = %SETUP_DIR%
parent_version_no = 0.0
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 -I -t600 -Usa -Phitachi -w1000 -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 =      1507
global_flag = 0
sequence_no =      2 step_level =      1      parent_step_id =      1466 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 =      %SETUP_DIR%
parent_version_no = 0.0
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 -Phitachi -w1000 -d %DBNAME% -e -i
%SETUP_DIR%\%showplan_queries%SP_TEXT_%QUERY%.sql -w 4096 -o
%SETUP_DIR%\%showplan_Out%\%SP_TEXT_%QUERY%.out

```

S%PARALLEL\_PROCESS%

*step\_label* = Generate Flat Files to FlatFile\_Dir\_1 *step\_id* = 1508  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
= -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_1%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for REGION and NATION Tables  
%TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%dbgen -T s -fF -q -b  
%TOOLS\_DIR%\%dists.dss -  
s%SCALEFACTOR% -C%TOTAL\_LOAD\_FILES\_PER\_TABLE% -

```

step_label = Generate Flat Files to FlatFile_Dir_2          step_id = 1509
global_flag = 0

sequence_no = 2 step_level = 2 parent_step_id = 1467 enabled_flag
= -1

iterator_name = PARALLEL_PROCESS degree_parallelism 1
execution_mechanism = 2 continuation_criteria = 2
failure_details =

step_file_name =
version_no = 0.0

start_directory = %FLATFILE_DIR_2%
parent_version_no = 0.0

step_text = :: Generate Data for CUSTOMER Table
             %TOOLS_DIR%\%DBGen%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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* = 1510  
*global\_flag* = 0  
*sequence\_no* = 3 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_3%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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* = 1511  
*global\_flag* = 0  
*sequence\_no* = 4 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_4%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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\_5 *step\_id* = 1512  
*global\_flag* = 0  
*sequence\_no* = 5 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_5%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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* = 1513  
*global\_flag* = 0  
*sequence\_no* = 6 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_6%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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* = 1514  
*global\_flag* = 0  
*sequence\_no* = 7 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_7%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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* = 1515  
*global\_flag* = 0  
*sequence\_no* = 8 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_8%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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_9          step_id = 1516
global_flag = 0
sequence_no = 9 step_level = 2 parent_step_id = 1467 enabled_flag
= -1
iterator_name = PARALLEL_PROCESS degree_parallelism 1
execution_mechanism = 2 continuation_criteria = 2
failure_details =
step_file_name =
version_no = 0.0
start_directory = %FLATFILE_DIR_9%
parent_version_no = 0.0
step_text = :: Generate Data for CUSTOMER Table
              %TOOLS_DIR%\%DBGen%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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 = 1517
global_flag = 0

sequence_no = 10 step_level = 2      parent_step_id = 1467 enabled_flag
= -1

iterator_name = PARALLEL_PROCESS      degree_parallelism 1
execution_mechanism = 2              continuation_criteria = 2
failure_details =

step_file_name =
version_no = 0.0

start_directory = %FLATFILE_DIR_10%
parent_version_no = 0.0

step_text = :: Generate Data for CUSTOMER Table
              %TOOLS_DIR%\%DBGen%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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 = 1518
global_flag = 0
sequence_no = 11 step_level = 2 parent_step_id = 1467 enabled_flag
= 0
iterator_name = PARALLEL_PROCESS degree_parallelism 4
execution_mechanism = 2 continuation_criteria = 2
failure_details =
step_file_name =
version_no = 0.0
start_directory = %FLATFILE_DIR_11%
parent_version_no = 0.0
step_text = :: Generate Data for CUSTOMER Table
              %TOOLS_DIR%\%DBGen%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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* = 1519  
*global\_flag* = 0  
*sequence\_no* = 12 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = 0  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 4  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_12%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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 = 1520
global_flag = 0
sequence_no = 13 step_level = 2      parent_step_id = 1467 enabled_flag
= 0
iterator_name = PARALLEL_PROCESS      degree_parallelism 4
execution_mechanism = 2      continuation_criteria = 2
failure_details =
step_file_name =
version_no = 0.0
start_directory = %FLATFILE_DIR_13%
parent_version_no = 0.0

step_text = :: Generate Data for CUSTOMER Table
              %TOOLS_DIR%\%DBGen%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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* = 1521  
*global\_flag* = 0  
*sequence\_no* = 14 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = 0  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 4  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_14%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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 = 1522
global_flag = 0

sequence_no = 15 step_level = 2      parent_step_id = 1467 enabled_flag
= 0

iterator_name = PARALLEL_PROCESS      degree_parallelism 4
execution_mechanism = 2              continuation_criteria = 2
failure_details =

step_file_name =
version_no = 0.0

start_directory = %FLATFILE_DIR_15%
parent_version_no = 0.0

step_text = :: Generate Data for CUSTOMER Table
              %TOOLS_DIR%\%DBGen%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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%\%SYSTEM_ARCHITECTURE%\%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* = 1523  
*global\_flag* = 0  
*sequence\_no* = 16 *step\_level* = 2 *parent\_step\_id* = 1467 *enabled\_flag*  
 = 0  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 4  
*execution\_mechanism* = 2 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = %FLATFILE\_DIR\_16%  
*parent\_version\_no* = 0.0  
*step\_text* = :: Generate Data for CUSTOMER Table  
 %TOOLS\_DIR%\%DBGen%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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%\%SYSTEM\_ARCHITECTURE%\%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* = 1524  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 2 *parent\_step\_id* = 1476 *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* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_DB  
*parent\_version\_no* = 0.0  
*step\_text* =

<i>step_label</i> = Create Base Tables	<i>step_id</i> = 1525	<i>iterator_name</i> = %FLAT_FILE_PARALLELISM%	<i>degree_parallelism</i>
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 2	<i>step_level</i> = 2	<i>parent_step_id</i> = 1476	<i>enabled_flag</i>
= -1		<i>failure_details</i> =	
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>step_file_name</i> =	
<i>execution_mechanism</i> = 1	<i>continuation_criteria</i> = 1	<i>version_no</i> =	0.0
<i>failure_details</i> =		<i>start_directory</i> =	
<i>step_file_name</i> = %SETUP_DIR%¥%DBNAME%¥CreateTables.sql		<i>parent_version_no</i> =	0.0
<i>version_no</i> = 0.0		<i>step_text</i> =	
<i>start_directory</i> = Dynamic_Connection_to_DB		<i>step_label</i> = Files 3	<i>step_id</i> = 1528
<i>parent_version_no</i> = 0.0		<i>global_flag</i> = 0	
<i>step_text</i> =		<i>sequence_no</i> = 3	<i>step_level</i> = 2
		= -1	<i>parent_step_id</i> = 1477
			<i>enabled_flag</i>
<i>step_label</i> = Files 1	<i>step_id</i> = 1526	<i>iterator_name</i> = %FLAT_FILE_PARALLELISM%	<i>degree_parallelism</i>
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 1	<i>step_level</i> = 2	<i>parent_step_id</i> = 1477	<i>enabled_flag</i>
= -1		<i>failure_details</i> =	
<i>iterator_name</i> =	<i>degree_parallelism</i>	<i>step_file_name</i> =	
%FLAT_FILE_PARALLELISM%		<i>version_no</i> =	0.0
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	
<i>failure_details</i> =		<i>parent_version_no</i> =	0.0
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Files 2	<i>step_id</i> = 1527		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 2	<i>step_level</i> = 2	<i>parent_step_id</i> = 1477	<i>enabled_flag</i>
= -1			

				=	-1					
<i>step_label</i> = Files 4			<i>step_id</i> = 1529			<i>iterator_name</i> =			<i>degree_parallelism</i>	
<i>global_flag</i> = 0						%FLAT_FILE_PARALLELISM%				
<i>sequence_no</i> = 4	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>		<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0	
=	-1					<i>failure_details</i> =				
<i>iterator_name</i> =			<i>degree_parallelism</i>			<i>step_file_name</i> =				
%FLAT_FILE_PARALLELISM%						<i>version_no</i> =				0.0
<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0			<i>start_directory</i> =				
<i>failure_details</i> =						<i>parent_version_no</i> =				0.0
<i>step_file_name</i> =						<i>step_text</i> =				
<i>version_no</i> =										
<i>start_directory</i> =										
<i>parent_version_no</i> =						<i>step_label</i> = Files 7			<i>step_id</i> = 1532	
<i>step_text</i> =						<i>global_flag</i> = 0				
						<i>sequence_no</i> = 7	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>
						=	-1			
<i>step_label</i> = Files 5			<i>step_id</i> = 1530			<i>iterator_name</i> =			<i>degree_parallelism</i>	
<i>global_flag</i> = 0						%FLAT_FILE_PARALLELISM%				
<i>sequence_no</i> = 5	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>		<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0	
=	-1					<i>failure_details</i> =				
<i>iterator_name</i> =			<i>degree_parallelism</i>			<i>step_file_name</i> =				
%FLAT_FILE_PARALLELISM%						<i>version_no</i> =				0.0
<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0			<i>start_directory</i> =				
<i>failure_details</i> =						<i>parent_version_no</i> =				0.0
<i>step_file_name</i> =						<i>step_text</i> =				
<i>version_no</i> =										
<i>start_directory</i> =										
<i>parent_version_no</i> =										
<i>step_text</i> =										
<i>step_label</i> = Files 6			<i>step_id</i> = 1531							
<i>global_flag</i> = 0										
<i>sequence_no</i> = 6	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>						

---

				=	-1					
<i>step_label</i> = Files 8			<i>step_id</i> = 1533	<i>iterator_name</i> =		<i>degree_parallelism</i>				
<i>global_flag</i> = 0				%FLAT_FILE_PARALLELISM%						
<i>sequence_no</i> = 8	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>	<i>execution_mechanism</i> = 0		<i>continuation_criteria</i> = 0			
=	-1				<i>failure_details</i> =					
<i>iterator_name</i> =			<i>degree_parallelism</i>		<i>step_file_name</i> =					0.0
%FLAT_FILE_PARALLELISM%					<i>version_no</i> =					
<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0		<i>start_directory</i> =					0.0
<i>failure_details</i> =					<i>parent_version_no</i> =					
<i>step_file_name</i> =					<i>step_text</i> =					
<i>version_no</i> =			0.0							
<i>start_directory</i> =					<i>step_label</i> = Files 11		<i>step_id</i> = 1536			
<i>parent_version_no</i> =			0.0		<i>global_flag</i> = 0					
<i>step_text</i> =					<i>sequence_no</i> = 11	<i>step_level</i> = 2	<i>parent_step_id</i> = 1477	<i>enabled_flag</i>		
					=	-1				
<i>step_label</i> = Files 9			<i>step_id</i> = 1534	<i>iterator_name</i> =		<i>degree_parallelism</i>				
<i>global_flag</i> = 0				%FLAT_FILE_PARALLELISM%						
<i>sequence_no</i> = 9	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>	<i>execution_mechanism</i> = 0		<i>continuation_criteria</i> = 0			
=	-1				<i>failure_details</i> =					
<i>iterator_name</i> =			<i>degree_parallelism</i>		<i>step_file_name</i> =					0.0
%FLAT_FILE_PARALLELISM%					<i>version_no</i> =					
<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0		<i>start_directory</i> =					0.0
<i>failure_details</i> =					<i>parent_version_no</i> =					
<i>step_file_name</i> =					<i>step_text</i> =					
<i>version_no</i> =			0.0							
<i>start_directory</i> =										
<i>parent_version_no</i> =			0.0							
<i>step_text</i> =										
<i>step_label</i> = Files 10			<i>step_id</i> = 1535							
<i>global_flag</i> = 0										
<i>sequence_no</i> = 10	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>						

---

				=	-1					
<i>step_label</i> = Files 12			<i>step_id</i> = 1537			<i>iterator_name</i> =			<i>degree_parallelism</i>	
<i>global_flag</i> = 0						%FLAT_FILE_PARALLELISM%				
<i>sequence_no</i> = 12	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>		<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0	
=	-1					<i>failure_details</i> =				
<i>iterator_name</i> =			<i>degree_parallelism</i>			<i>step_file_name</i> =				0.0
%FLAT_FILE_PARALLELISM%						<i>version_no</i> =				
<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0			<i>start_directory</i> =				
<i>failure_details</i> =						<i>parent_version_no</i> =			0.0	
<i>step_file_name</i> =						<i>step_text</i> =				
<i>version_no</i> =					0.0					
<i>start_directory</i> =						<i>step_label</i> = Files 15			<i>step_id</i> = 1540	
<i>parent_version_no</i> =					0.0	<i>global_flag</i> = 0				
<i>step_text</i> =						<i>sequence_no</i> = 15	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>
						=	-1			
<i>step_label</i> = Files 13			<i>step_id</i> = 1538			<i>iterator_name</i> =			<i>degree_parallelism</i>	
<i>global_flag</i> = 0						%FLAT_FILE_PARALLELISM%				
<i>sequence_no</i> = 13	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>		<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0	
=	-1					<i>failure_details</i> =				
<i>iterator_name</i> =			<i>degree_parallelism</i>			<i>step_file_name</i> =				0.0
%FLAT_FILE_PARALLELISM%						<i>version_no</i> =				
<i>execution_mechanism</i> = 0			<i>continuation_criteria</i> = 0			<i>start_directory</i> =				
<i>failure_details</i> =						<i>parent_version_no</i> =			0.0	
<i>step_file_name</i> =						<i>step_text</i> =				
<i>version_no</i> =					0.0					
<i>start_directory</i> =										
<i>parent_version_no</i> =					0.0					
<i>step_text</i> =										
<i>step_label</i> = Files 14			<i>step_id</i> = 1539							
<i>global_flag</i> = 0										
<i>sequence_no</i> = 14	<i>step_level</i> = 2		<i>parent_step_id</i> = 1477	<i>enabled_flag</i>						

---

```

step_label = Files 16                                step_id = 1541
global_flag = 0
sequence_no = 16 step_level = 2 parent_step_id = 1477 enabled_flag
= -1
iterator_name = degree_parallelism
%FLAT_FILE_PARALLELISM%
execution_mechanism = 0 continuation_criteria = 0
failure_details =
step_file_name =
version_no = 0.0
start_directory =
parent_version_no = 0.0
step_text =

global_flag = 0
sequence_no = 2 step_level = 2 parent_step_id = 1478 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 = 0.0
step_text = bulk insert %DBNAME%..REGION
from '%FLATFILE_DIR_1%¥region.tbl'
with (FieldTerminator = '|', RowTerminator =|¥n',tablock)

```

```

step_label = Load Nation Table                        step_id = 1542
global_flag = 0
sequence_no = 1 step_level = 2 parent_step_id = 1478 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 = 0.0
step_text = bulk insert %DBNAME%..NATION
from '%FLATFILE_DIR_1%¥nation.tbl'
with (FieldTerminator = '|', RowTerminator =|¥n',tablock)

```

```

step_label = Load Region Table                       step_id = 1543

```

---

*step\_label* = Install RF1 Stored Procedure(s) *step\_id* = 1544  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 2 *parent\_step\_id* = 1485 *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* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_DB  
*parent\_version\_no* = 0.0  
*step\_text* =

*step\_label* = Install RF2 Stored Procedure(s) *step\_id* = 1545  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 2 *parent\_step\_id* = 1485 *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* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_DB  
*parent\_version\_no* = 0.0  
*step\_text* =

*step\_label* = Create Backup Device(s) (For ACID) *step\_id* = 1546  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 2 *parent\_step\_id* = 1486 *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* = 0.0  
*step\_text* =

*step\_label* = Execute the backup (For ACID) *step\_id* = 1547  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 2 *parent\_step\_id* = 1486 *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* = 0.0  
*step\_text* =



```

step_label = Generate Power Query Directory                                step_id = 1548
global_flag = 0
sequence_no = 1 step_level = 2 parent_step_id = 1501 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 = 0.0
step_text = if not exist %QUERY_DIR%\Power mkdir %QUERY_DIR%\Power

step_label = Parallel Power Query Generation                                step_id = 1550
global_flag = 0
sequence_no = 3 step_level = 2 parent_step_id = 1501 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 = 0.0
step_text =

```

```

step_label = Generate QGen Command File                                step_id = 1549
global_flag = 0
sequence_no = 2 step_level = 2 parent_step_id = 1501 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 = %TEMPLATE_DIR%
parent_version_no = 0.0
step_text = ::
:: First use OSQL to grab the QgenSeed value from TPCH_AUX_TABLE
::
osql -Usa -P%saPASSWORD% -n -d%DBNAME% -w 255 -
i%SETUP_DIR%\%Generate%GenQGENcmd.sql >
%SETUP_DIR%\%Generate%QgenCmd.cmd

```

```

step_label = Set Seed Value for Stream                                step_id = 1551
global_flag = 0

sequence_no = 1 step_level = 2 parent_step_id = 1502 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 = 0.0

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 = 1552
global_flag = 0

sequence_no = 2 step_level = 2 parent_step_id = 1502 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 = 0.0

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 = 1553
global_flag = 0

sequence_no = 3 step_level = 2 parent_step_id = 1502 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 = %TEMPLATE_DIR%
parent_version_no = 0.0

step_text = ::
:: First use OSQL to grab the QgenSeed value from TPCH_AUX_TABLE
::
osql -Usa -P%saPASSWORD% -n -d%DBNAME% -w 255 -
i%SETUP_DIR%¥Generate¥GenQgenStreamCmd.sql >
%SETUP_DIR%¥Generate¥QgenStreamCmd.cmd

```

```

step_label = Parallel Stream Query Generation                                step_id = 1554
global_flag = 0
sequence_no = 4 step_level = 2 parent_step_id = 1502 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 = 0.0
step_text =

step_label = Generate Validation Queries                                step_id = 1556
global_flag = 0
sequence_no = 1 step_level = 2 parent_step_id = 1503 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 = Re-set QGen Seed Value                                step_id = 1555
global_flag = 0
sequence_no = 5 step_level = 2 parent_step_id = 1502 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 = --
-- Resets the QGen seed kept in the temporary table
--
UPDATE TPCH_AUX_TABLE
SET QgenSeed = QgenSeed - %STREAM_NUM%

```

*step\_label* = Validation Query 1 *step\_id* = 1557  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1558  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1559  
*global\_flag* = 0  
*sequence\_no* = 3 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1560  
*global\_flag* = 0  
*sequence\_no* = 4 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1561  
*global\_flag* = 0  
*sequence\_no* = 5 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1562  
*global\_flag* = 0  
*sequence\_no* = 6 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1563  
*global\_flag* = 0  
*sequence\_no* = 7 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1564  
*global\_flag* = 0  
*sequence\_no* = 8 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1565  
*global\_flag* = 0  
*sequence\_no* = 9 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1566  
*global\_flag* = 0  
*sequence\_no* = 10 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1567  
*global\_flag* = 0  
*sequence\_no* = 11 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1568  
*global\_flag* = 0  
*sequence\_no* = 12 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1569  
*global\_flag* = 0  
*sequence\_no* = 13 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1570  
*global\_flag* = 0  
*sequence\_no* = 14 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1571  
*global\_flag* = 0  
*sequence\_no* = 15 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1572  
*global\_flag* = 0  
*sequence\_no* = 16 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1573  
*global\_flag* = 0  
*sequence\_no* = 17 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1574  
*global\_flag* = 0  
*sequence\_no* = 18 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1575  
*global\_flag* = 0  
*sequence\_no* = 19 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1576  
*global\_flag* = 0  
*sequence\_no* = 20 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1577  
*global\_flag* = 0  
*sequence\_no* = 21 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1578  
*global\_flag* = 0  
*sequence\_no* = 22 *step\_level* = 2 *parent\_step\_id* = 1504 *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* = 1579  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1550 *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* = 1580  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1554 *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 = Load Partitioned Tables from FlatFile_Dir_1          step_id = 1581
global_flag = 0
sequence_no = 1 step_level = 3 parent_step_id = 1526 enabled_flag
= -1
iterator_name = PARALLEL_PROCESS degree_parallelism 1
execution_mechanism = 1 continuation_criteria = 2
failure_details =
step_file_name =
version_no = 0.0
start_directory = Dynamic_Connection_to_Master
parent_version_no = 0.0
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_3          step_id = 1583
global_flag = 0
sequence_no = 1 step_level = 3 parent_step_id = 1528 enabled_flag
= -1
iterator_name = PARALLEL_PROCESS degree_parallelism 1
execution_mechanism = 1 continuation_criteria = 2
failure_details =
step_file_name =
version_no = 0.0
start_directory = Dynamic_Connection_to_Master
parent_version_no = 0.0
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_2          step_id = 1582
global_flag = 0
sequence_no = 1 step_level = 3 parent_step_id = 1527 enabled_flag
= -1
iterator_name = PARALLEL_PROCESS degree_parallelism 1
execution_mechanism = 1 continuation_criteria = 2
failure_details =
step_file_name =
version_no = 0.0
start_directory = Dynamic_Connection_to_Master
parent_version_no = 0.0
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\_4 *step\_id* = 1584  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1529 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_6 *step\_id* = 1586  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1531 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_5 *step\_id* = 1585  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1530 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_7 *step\_id* = 1587  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1532 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_9 *step\_id* = 1589  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1534 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_8 *step\_id* = 1588  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1533 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_10 *step\_id* = 1590  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1535 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 1  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_12 *step\_id* = 1592  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1537 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 4  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_11 *step\_id* = 1591  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1536 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 4  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_13 *step\_id* = 1593  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1538 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 4  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_15 *step\_id* = 1595  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1540 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 4  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*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\_14 *step\_id* = 1594  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1539 *enabled\_flag*  
 = -1  
*iterator\_name* = PARALLEL\_PROCESS *degree\_parallelism* 4  
*execution\_mechanism* = 1 *continuation\_criteria* = 2  
*failure\_details* =  
*step\_file\_name* =  
*version\_no* = 0.0  
*start\_directory* = Dynamic\_Connection\_to\_Master  
*parent\_version\_no* = 0.0  
*step\_text* = bulk insert %DBNAME%..%TABLE%  
 from '%FLATFILE\_DIR\_14%¥%%TABLE%.tbl.%PARALLEL\_PROCESS%'  
 with (FieldTerminator = '|', RowTerminator  
 ='|¥n',tablock,ROWS\_PER\_BATCH=%BCP\_ROWS%)

14 63 1473 0.0 2 1452 0.0

```

step_label = Load Partitioned Tables from FlatFile_Dir_16      step_id = 1596
global_flag = 0
sequence_no = 1 step_level = 3      parent_step_id = 1541 enabled_flag
= -1
iterator_name = PARALLEL_PROCESS      degree_parallelism 4
execution_mechanism = 1      continuation_criteria = 2
failure_details =
step_file_name =
version_no = 0.0
start_directory = Dynamic_Connection_to_Master
parent_version_no = 0.0
step_text = bulk insert %DBNAME%..%TABLE%
            from '%FLATFILE_DIR_16%¥%%TABLE%.tbl.%PARALLEL_PROCESS%'
            with (FieldTerminator = '|', RowTerminator
            ='|¥n',tablock,ROWS_PER_BATCH=%BCP_ROWS%)

```

## Iterator\_values

workspace_id	step_id	version_no	type	iterator_value
14	1586	0.0	1	6
14	1581	0.0	1	1
14	1582	0.0	3	1
14	1582	0.0	2	2
14	1582	0.0	1	2
14	1583	0.0	3	1

## Step\_constraints

workspace_id	constraint_id	step_id	version_no	constraint_global_id	constraint_type	constraint_step_id	constraint_on_no	constraint_no
14	70	1487	0.0	1	1454	0.0	2	2
14	69	1473	0.0	2	1454	0.0	2	2
14	68	1469	0.0	2	1454	0.0	0	0
14	67	1472	0.0	2	1454	0.0	0	0
14	66	1487	0.0	1	1453	0.0	0	0
14	65	1473	0.0	2	1453	0.0	0	0
14	64	1487	0.0	1	1452	0.0	1	1

14	1583	0.0	1 3	014	1522	0.0	2 60
14	1583	0.0	2 3	014	1517	0.0	2 10
14	1584	0.0	3 1	014	1517	0.0	1 10
14	1584	0.0	2 4	014	1518	0.0	1 41
14	1584	0.0	1 4	014	1518	0.0	2 44
14	1585	0.0	3 1	014	1518	0.0	3 1
14	1585	0.0	1 5	014	1519	0.0	3 1
14	1517	0.0	3 1	0			
14	1586	0.0	3 1	0			
14	1580	0.0	3 1	0			
14	1586	0.0	2 6	0			
14	1587	0.0	3 1	0			
14	1587	0.0	2 7	0			
14	1587	0.0	1 7	0			
14	1588	0.0	2 8	0			
14	1588	0.0	1 8	0			
14	1588	0.0	3 1	0			
14	1589	0.0	3 1	0			
14	1589	0.0	2 9	0			
14	1589	0.0	1 9	0			
14	1590	0.0	3 1	0			
14	1590	0.0	2 10	0			
14	1590	0.0	1 10	0			
14	1585	0.0	2 5	0			



14	1519	0.0	2 48	014	1512	0.0	1 5
14	1519	0.0	1 45	014	1591	0.0	1 11
14	1520	0.0	1 49	014	1507	0.0	1 1
14	1520	0.0	3 1	014	1507	0.0	2 22
14	1520	0.0	2 52	014	1508	0.0	1 1
14	1521	0.0	1 53	014	1508	0.0	3 1
14	1521	0.0	3 1	014	1508	0.0	2 1
14	1581	0.0	2 1	0			
14	1556	0.0	3 1	0			
14	1591	0.0	3 1	0			
14	1580	0.0	2 22	0			
14	1580	0.0	1 1	0			
14	1579	0.0	1 1	0			
14	1579	0.0	2 22	0			
14	1521	0.0	2 56	0			
14	1556	0.0	1 1	0			
14	1522	0.0	3 1	0			
14	1556	0.0	2 22	0			
14	1523	0.0	3 1	0			
14	1523	0.0	2 64	0			
14	1523	0.0	1 61	0			
14	1522	0.0	1 57	0			
14	1581	0.0	3 1	0			
14	1579	0.0	3 1	0			

14	1509	0.0	1 2	014	1596	0.0	2 16
14	1509	0.0	2 2	014	1516	0.0	2 9
14	1509	0.0	3 1	014	1592	0.0	1 12
14	1510	0.0	1 3	014	1592	0.0	2 12
14	1510	0.0	2 3	014	1592	0.0	3 1
14	1510	0.0	3 1	014	1593	0.0	3 1
14	1511	0.0	3 1	014	1593	0.0	1 13
14	1506	0.0	3 1	0			
14	1514	0.0	1 7	0			
14	1516	0.0	3 1	0			
14	1516	0.0	1 9	0			
14	1515	0.0	2 8	0			
14	1515	0.0	1 8	0			
14	1515	0.0	3 1	0			
14	1511	0.0	2 4	0			
14	1514	0.0	2 7	0			
14	1511	0.0	1 4	0			
14	1513	0.0	1 6	0			
14	1513	0.0	2 6	0			
14	1513	0.0	3 1	0			
14	1512	0.0	3 1	0			
14	1512	0.0	2 5	0			
14	1506	0.0	2 22	0			
14	1514	0.0	3 1	0			

14	1593	0.0	2 13	0
14	1594	0.0	1 14	0
14	1594	0.0	3 1	0
14	1594	0.0	2 14	0
14	1595	0.0	1 15	0
14	1595	0.0	2 15	0
14	1595	0.0	3 1	0
14	1507	0.0	3 1	0
14	1477	0.0	4 LINEITEM	0
14	1506	0.0	1 1	0
14	1505	0.0	3 1	0
14	1505	0.0	2 22	0
14	1505	0.0	1 1	0
14	1502	0.0	1 1	0
14	1596	0.0	3 1	0
14	1502	0.0	2 %MAX_STREAMS%	0
14	1596	0.0	1 16	0
14	1477	0.0	4 ORDERS	1
14	1477	0.0	4 SUPPLIER	2
14	1477	0.0	4 PART	3
14	1477	0.0	4 CUSTOMER	4
14	1477	0.0	4 PARTSUPP	5
14	1591	0.0	2 11	0
14	1502	0.0	3 1	0

## *Workspace\_parameters*

<i>workspace_id</i>	15	<i>parameter_id =</i>	264	<i>parameter_type =</i>	0
<i>parameter_name</i>			TOOLS_DIR		
<i>parameter_value</i>			%KIT_DIR%\Tools		
<i>workspace_id</i>	15	<i>parameter_id =</i>	255	<i>parameter_type =</i>	3
<i>parameter_name</i>			OUTPUT_DIR		
<i>parameter_value</i>			C:\MSTPCH-1.200\output\95		
<i>workspace_id</i>	15	<i>parameter_id =</i>	256	<i>parameter_type =</i>	0
<i>parameter_name</i>			QUERY_DIR		
<i>parameter_value</i>			%RUN_DIR%\Queries		
<i>workspace_id</i>	15	<i>parameter_id =</i>	257	<i>parameter_type =</i>	0
<i>parameter_name</i>			DBNAME		
<i>parameter_value</i>			tpch300g		
<i>workspace_id</i>	15	<i>parameter_id =</i>	258	<i>parameter_type =</i>	0
<i>parameter_name</i>			DELETE_PARALLELISM		
<i>parameter_value</i>			24		
<i>workspace_id</i>	15	<i>parameter_id =</i>	259	<i>parameter_type =</i>	0
<i>parameter_name</i>			INSERT_PARALLELISM		
<i>parameter_value</i>			48		
<i>workspace_id</i>	15	<i>parameter_id =</i>	260	<i>parameter_type =</i>	0
<i>parameter_name</i>			DELETE_EXECUTIONS		
<i>parameter_value</i>			72		
<i>workspace_id</i>	15	<i>parameter_id =</i>	261	<i>parameter_type =</i>	0
<i>parameter_name</i>			INSERT_EXECUTIONS		

*workspace\_id*  
*parameter\_name*  
*parameter\_value*

15 *parameter\_id =*  
RUN\_DIR  
%KIT\_DIR%\Run

254 *parameter\_type =*

0

<i>workspace_id</i>	15	<i>parameter_id =</i>	263	<i>parameter_type =</i>	3	<i>parameter_value</i>	16		
<i>parameter_name</i>		DEFAULT_DIR							
<i>parameter_value</i>		%KIT_DIR%¥output							
<i>workspace_id</i>	15	<i>parameter_id =</i>	273	<i>parameter_type =</i>	0				
<i>parameter_name</i>		SERVERNAME							
<i>parameter_value</i>		EBS64.							
<i>workspace_id</i>	15	<i>parameter_id =</i>	265	<i>parameter_type =</i>	0				
<i>parameter_name</i>		BATCH_SIZE							
<i>parameter_value</i>		10							
<i>workspace_id</i>	15	<i>parameter_id =</i>	266	<i>parameter_type =</i>	0				
<i>parameter_name</i>		KIT_DIR							
<i>parameter_value</i>		C:¥MSTPCH_Beta.200							
<i>workspace_id</i>	15	<i>parameter_id =</i>	267	<i>parameter_type =</i>	3				
<i>parameter_name</i>		RUN_ID							
<i>parameter_value</i>		95							
<i>workspace_id</i>	15	<i>parameter_id =</i>	268	<i>parameter_type =</i>	0				
<i>parameter_name</i>		TRACEFLAGS							
<i>parameter_value</i>		-c -x -E -T834 -T2301							
<i>workspace_id</i>	15	<i>parameter_id =</i>	269	<i>parameter_type =</i>	0				
<i>parameter_name</i>		SETUP_DIR							
<i>parameter_value</i>		%KIT_DIR%¥Setup							
<i>workspace_id</i>	15	<i>parameter_id =</i>	270	<i>parameter_type =</i>	0				
<i>parameter_name</i>		RF_FLATFILE_DIR							
<i>parameter_value</i>		E:¥mnt¥fs¥14¥RF_Flat_File_16							
<i>workspace_id</i>	15	<i>parameter_id =</i>	271	<i>parameter_type =</i>	0				
<i>parameter_name</i>		FILES_PER_UPDATE_SET							

*workspace\_id* 15 *parameter\_id =* 262 *parameter\_type =* 0  
*parameter\_name* MAX\_STREAMS  
*parameter\_value* 6

## *Connection\_dtls*

<i>worksp</i>	<i>connection</i>	<i>connection_name</i>	<i>connecti</i>	<i>on_type</i>
<i>connection_string_name</i>	<i>ace_id</i>	<i>_name_id</i>		
	15	114	Dynamic_Connection_to_Master	MASTERCONNECTION 2
	15	113	Power_Dynamic_DB_Connection	DBCONECTION 2
	15	112	Throughput_Static_Stream_10_Connection	DBCONECTION 1
	15	111	Throughput_Static_Stream_9_Connection	DBCONECTION 1
	15	110	Throughput_Static_Stream_8_Connection	DBCONECTION 1
	15	109	Throughput_Static_Stream_7_Connection	DBCONECTION 1
	15	108	Throughput_Static_Stream_6_Connection	DBCONECTION 1
	15	107	Throughput_Static_Stream_5_Connection	DBCONECTION 1
	15	106	Throughput_Static_Stream_4_Connection	DBCONECTION 1
	15	105	Throughput_Static_Stream_3_Connection	DBCONECTION 1
	15	104	Throughput_Static_Stream_2_Connection	DBCONECTION 1
	15	103	Throughput_Static_Stream_1_Connection	DBCONECTION 1
	15	102	Throughput_RF_Connection	DBCONECTION 2
	15	101	Dynamic_Connection_to_DB	DBCONECTION 2
	15	100	Power_Static_DB_Connection	DBCONECTION 1

## Workspace\_connections

<i>workspace_id</i>	15
<i>connection_id</i>	29
<i>connection_name</i>	MASTERCONNECTION
<i>connection_value</i>	DRIVER=SQL Native Client;SERVER=%SERVERNAME%;UID=sa;PWD=%saPASSWORD%;
<i>description</i>	
<i>no_count_display</i>	0
<i>no_execute</i>	0
<i>parse_query_only</i>	0
<i>ANSI_quoted_identifiers</i>	0
<i>ANSI_nulls</i>	-1
<i>show_query_plan</i>	0
<i>show_stats_time</i>	0
<i>show_stats_io</i>	0
<i>parse_odbc_msg_prefixes</i>	-1
<i>row_count</i>	0
<i>tsql_batch_separator</i>	GO
<i>query_time_out</i>	0
<i>server_language</i>	(Default)
<i>character_translation</i>	-1
<i>regional_settings</i>	0
<i>workspace_id</i>	15
<i>connection_id</i>	28
<i>connection_name</i>	DBCONNECTION
<i>connection_value</i>	DRIVER=SQL Native
<i>description</i>	
<i>no_count_display</i>	0
<i>no_execute</i>	0
<i>parse_query_only</i>	0
<i>ANSI_quoted_identifiers</i>	-1
<i>ANSI_nulls</i>	-1

<i>show_query_plan</i>	0
<i>show_stats_time</i>	0
<i>show_stats_io</i>	0
<i>parse_odbc_msg_prefixes</i>	-1
<i>row_count</i>	0
<i>tsql_batch_separator</i>	GO
<i>query_time_out</i>	0
<i>server_language</i>	(Default)

*character\_translation* = -1  
*regional\_settings* = 0

## Att\_steps

*workspace\_id* = 15

*step\_label* = SqlServer Startup *step\_id* = 1597  
*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.exe -c %TRACEFLAGS%  
%TOOLS\_DIR%\Utility\wait4sql 40000

*step\_label* = SqlServer Shutdown *step\_id* = 1598  
*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* = isql -Usa -P%saPASSWORD% -t10 -Q"shutdown"  
%TOOLS\_DIR%\Utility\wait4sql 10000

*step\_label* = Wait For SQL Server *step\_id* = 1599  
*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 = Clear any Outstanding Semaphores
global_flag = 0
sequence_no = 1 step_level = 0 parent_step_id = 0 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 =
::
:: 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 Throughput Run
global_flag = 0
sequence_no = 3 step_level = 0 parent_step_id = 0 enabled_flag
= -1
iterator_name =
execution_mechanism = 0 degree_parallelism 2
failure_details = continuation_criteria = 0
step_file_name =
version_no = 0.0
start_directory =
parent_version_no = 0.0
step_text =

```

```

step_label = Execute Power Run
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 = 0.0
start_directory =
parent_version_no = 0.0
step_text =

```

<i>step_label</i> = Power - Execute RF1	<i>step_id</i> = 1603	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 1	<i>step_level</i> = 1	<i>parent_step_id</i> = 1601	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> =	0.0
<i>failure_details</i> =		<i>step_text</i> =	
<i>step_file_name</i> =		<i>step_label</i> = Power - Increment Update Set	<i>step_id</i> = 1606
<i>version_no</i> = 0.0		<i>global_flag</i> = 0	
<i>start_directory</i> =		<i>sequence_no</i> = 4	<i>step_level</i> = 1
<i>parent_version_no</i> = 0.0		= -1	<i>parent_step_id</i> = 1601
<i>step_text</i> =		<i>enabled_flag</i>	
<i>step_label</i> = Power - Sequential Query Execution	<i>step_id</i> = 1604	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 1	<i>continuation_criteria</i> = 2
<i>sequence_no</i> = 2	<i>step_level</i> = 1	<i>parent_step_id</i> = 1601	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	<i>version_no</i> = 0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>start_directory</i> = Dynamic_Connection_to_DB	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>parent_version_no</i> = 0.0	
<i>failure_details</i> =		<i>step_text</i> = UPDATE TPCH_AUX_TABLE SET updateset=updateset+1	
<i>step_file_name</i> =			
<i>version_no</i> = 0.0			
<i>start_directory</i> =			
<i>parent_version_no</i> = 0.0			
<i>step_text</i> =			
<i>step_label</i> = Power - Execute RF2	<i>step_id</i> = 1605		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 3	<i>step_level</i> = 1	<i>parent_step_id</i> = 1601	<i>enabled_flag</i>
= -1			

<i>step_label</i> = Sequential Refresh Stream Execution	<i>step_id</i> = 1607	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 1	<i>continuation_criteria</i> = 2
<i>sequence_no</i> = 1	<i>step_level</i> = 1	<i>parent_step_id</i> = 1602	<i>enabled_flag</i>
= -1		<i>failure_details</i> =	
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>step_file_name</i> = %QUERY_DIR%\¥Power¥14.sql	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>version_no</i> = 0.0	
<i>failure_details</i> =		<i>start_directory</i> = Power_Static_DB_Connection	
<i>step_file_name</i> =		<i>parent_version_no</i> = 0.0	
<i>version_no</i> =	0.0	<i>step_text</i> =	
<i>start_directory</i> =		<i>step_label</i> = Power - Execute Query 02	<i>step_id</i> = 1610
<i>parent_version_no</i> =	0.0	<i>global_flag</i> = 0	
<i>step_text</i> =		<i>sequence_no</i> = 2	<i>step_level</i> = 2
		= -1	<i>parent_step_id</i> = 1604
		<i>enabled_flag</i>	
		<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
		<i>execution_mechanism</i> = 1	<i>continuation_criteria</i> = 2
		<i>failure_details</i> =	
<i>step_label</i> = Parallel Stream Execution	<i>step_id</i> = 1608	<i>step_file_name</i> = %QUERY_DIR%\¥Power¥2.sql	
<i>global_flag</i> = 0		<i>version_no</i> = 0.0	
<i>sequence_no</i> = 2	<i>step_level</i> = 1	<i>start_directory</i> = Power_Static_DB_Connection	
= -1		<i>parent_version_no</i> = 0.0	
<i>iterator_name</i> =	<i>degree_parallelism</i> = %MAX_STREAMS%	<i>step_text</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0		
<i>failure_details</i> =			
<i>step_file_name</i> =			
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Power - Execute Query 14	<i>step_id</i> = 1609		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 1	<i>step_level</i> = 2	<i>parent_step_id</i> = 1604	<i>enabled_flag</i>
= -1			

*step\_label* = Power - Execute Query 09  
*global\_flag* = 0

*step\_id* = 1611

*sequence\_no* = 3 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power¥9.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 20  
*global\_flag* = 0

*step\_id* = 1612

*sequence\_no* = 4 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power¥20.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 06  
*global\_flag* = 0

*step\_id* = 1613

*sequence\_no* = 5 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power¥6.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 17

*step\_id* = 1614

*global\_flag* = 0

*sequence\_no* = 6 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power¥17.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

<p><i>step_label</i> = Power - Execute Query 18 <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 7 <i>step_level</i> = 2 = -1</p> <p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %QUERY_DIR%\Power¥18.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Power_Static_DB_Connection <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>step_id</i> = 1615</p> <p><i>parent_step_id</i> = 1604 <i>enabled_flag</i></p> <p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>	<p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %QUERY_DIR%\Power¥21.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Power_Static_DB_Connection <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>
<p><i>step_label</i> = Power - Execute Query 08 <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 8 <i>step_level</i> = 2 = -1</p> <p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %QUERY_DIR%\Power¥8.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Power_Static_DB_Connection <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>step_id</i> = 1616</p> <p><i>parent_step_id</i> = 1604 <i>enabled_flag</i></p> <p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>	<p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %QUERY_DIR%\Power¥13.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Power_Static_DB_Connection <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>
<p><i>step_label</i> = Power - Execute Query 21 <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 9 <i>step_level</i> = 2 = -1</p>	<p><i>step_id</i> = 1617</p> <p><i>parent_step_id</i> = 1604 <i>enabled_flag</i></p>		

<p><i>step_label</i> = Power - Execute Query 03 <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 11 <i>step_level</i> = 2 = -1</p> <p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %QUERY_DIR%\Power¥3.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Power_Static_DB_Connection <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>step_id</i> = 1619</p> <p><i>parent_step_id</i> = 1604 <i>enabled_flag</i></p> <p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>	<p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %QUERY_DIR%\Power¥16.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Power_Static_DB_Connection <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>
<p><i>step_label</i> = Power - Execute Query 22 <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 12 <i>step_level</i> = 2 = -1</p> <p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %QUERY_DIR%\Power¥22.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Power_Static_DB_Connection <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>step_id</i> = 1620</p> <p><i>parent_step_id</i> = 1604 <i>enabled_flag</i></p> <p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>	<p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %QUERY_DIR%\Power¥4.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Power_Static_DB_Connection <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>
<p><i>step_label</i> = Power - Execute Query 16 <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 13 <i>step_level</i> = 2 = -1</p>	<p><i>step_id</i> = 1621</p> <p><i>parent_step_id</i> = 1604 <i>enabled_flag</i></p>		

*step\_label* = Power - Execute Query 11  
*global\_flag* = 0

*step\_id* = 1623

*sequence\_no* = 15 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power¥11.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 15  
*global\_flag* = 0

*step\_id* = 1624

*sequence\_no* = 16 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power¥15.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 01  
*global\_flag* = 0

*step\_id* = 1625

*sequence\_no* = 17 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power¥1.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 10  
*global\_flag* = 0

*step\_id* = 1626

*sequence\_no* = 18 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power¥10.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 19  
*global\_flag* = 0

*step\_id* = 1627

*sequence\_no* = 19 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power\19.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 05  
*global\_flag* = 0

*step\_id* = 1628

*sequence\_no* = 20 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power\5.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 07  
*global\_flag* = 0

*step\_id* = 1629

*sequence\_no* = 21 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power\7.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =

*step\_label* = Power - Execute Query 12  
*global\_flag* = 0

*step\_id* = 1630

*sequence\_no* = 22 *step\_level* = 2  
= -1

*parent\_step\_id* = 1604 *enabled\_flag*

*iterator\_name* =

*degree\_parallelism* 1

*execution\_mechanism* = 1

*continuation\_criteria* = 2

*failure\_details* =

*step\_file\_name* = %QUERY\_DIR%\Power\12.sql  
*version\_no* = 0.0

*start\_directory* = Power\_Static\_DB\_Connection  
*parent\_version\_no* = 0.0

*step\_text* =



```

=
step_label = Throughput - Semaphore Loop for RF Delay          step_id = 1631
global_flag = 0
sequence_no = 1 step_level = 2 parent_step_id = 1607 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 -waitgroup S -count %MAX_STREAMS%

step_label = Throughput - Refresh Streams          step_id = 1632
global_flag = 0
sequence_no = 2 step_level = 2 parent_step_id = 1607 enabled_flag
= -1
iterator_name = STREAM_NUM 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 = Stream 1 Manager          step_id = 1633
global_flag = 0
sequence_no = 1 step_level = 2 parent_step_id = 1608 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 = Stream 2 Manager          step_id = 1634
global_flag = 0
sequence_no = 2 step_level = 2 parent_step_id = 1608 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 =

```

<i>step_label</i> = Stream 3 Manager	<i>step_id</i> = 1635	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 3	<i>step_level</i> = 2	<i>parent_step_id</i> = 1608	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0	<i>step_label</i> = Stream 6 Manager	<i>step_id</i> = 1638
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.0	<i>sequence_no</i> = 6	<i>step_level</i> = 2
<i>step_text</i> =		= -1	<i>parent_step_id</i> = 1608
		<i>enabled_flag</i>	
<i>step_label</i> = Stream 4 Manager	<i>step_id</i> = 1636	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 4	<i>step_level</i> = 2	<i>parent_step_id</i> = 1608	<i>enabled_flag</i>
= -1		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Stream 5 Manager	<i>step_id</i> = 1637		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 5	<i>step_level</i> = 2	<i>parent_step_id</i> = 1608	<i>enabled_flag</i>
= -1			

<i>step_label</i> = Stream 7 Manager	<i>step_id</i> = 1639	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 7	<i>step_level</i> = 2	<i>parent_step_id</i> = 1608	<i>enabled_flag</i>
= 0		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0	<i>step_label</i> = Stream 10 Manager	<i>step_id</i> = 1642
<i>start_directory</i> =		<i>global_flag</i> = 0	
<i>parent_version_no</i> =	0.0	<i>sequence_no</i> = 10	<i>step_level</i> = 2
<i>step_text</i> =		= 0	<i>parent_step_id</i> = 1608
		<i>enabled_flag</i>	
<i>step_label</i> = Stream 8 Manager	<i>step_id</i> = 1640	<i>iterator_name</i> =	<i>degree_parallelism</i> = 1
<i>global_flag</i> = 0		<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0
<i>sequence_no</i> = 8	<i>step_level</i> = 2	<i>parent_step_id</i> = 1608	<i>enabled_flag</i>
= 0		<i>step_file_name</i> =	0.0
<i>iterator_name</i> =	<i>degree_parallelism</i> = 1	<i>version_no</i> =	
<i>execution_mechanism</i> = 0	<i>continuation_criteria</i> = 0	<i>start_directory</i> =	0.0
<i>failure_details</i> =		<i>parent_version_no</i> =	
<i>step_file_name</i> =		<i>step_text</i> =	
<i>version_no</i> =	0.0		
<i>start_directory</i> =			
<i>parent_version_no</i> =	0.0		
<i>step_text</i> =			
<i>step_label</i> = Stream 9 Manager	<i>step_id</i> = 1641		
<i>global_flag</i> = 0			
<i>sequence_no</i> = 9	<i>step_level</i> = 2	<i>parent_step_id</i> = 1608	<i>enabled_flag</i>
= 0			

```

step_label = Create RF1 Tables                                step_id = 1643
global_flag = 0

sequence_no = 1 step_level = 2 parent_step_id = 1603 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 = 0.0

start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Parallel Load RF1 Input                          step_id = 1644
global_flag = 0

sequence_no = 2 step_level = 2 parent_step_id = 1603 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 = 0.0
step_text =

step_label = Create RF1 Indices                                step_id = 1645
global_flag = 0

sequence_no = 3 step_level = 2 parent_step_id = 1603 enabled_flag
= -1

iterator_name =
execution_mechanism = 1 continuation_criteria = 2
failure_details =

step_file_name = %SETUP_DIR%¥RF_Procs¥RF1_index.sql
version_no = 0.0

start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Parallel RF1 Execution                          step_id = 1646
global_flag = 0

sequence_no = 4 step_level = 2 parent_step_id = 1603 enabled_flag
= -1

iterator_name = degree_parallelism
%INSERT_PARALLELISM%
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 RF2 Tables                                step_id = 1647
global_flag = 0

sequence_no = 1 step_level = 2 parent_step_id = 1605 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 = 0.0

start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Parallel Load RF2 Input                          step_id = 1648
global_flag = 0

sequence_no = 2 step_level = 2 parent_step_id = 1605 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 = 0.0
step_text =

step_label = Create RF2 Indices                                step_id = 1649
global_flag = 0

sequence_no = 3 step_level = 2 parent_step_id = 1605 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 = 0.0

start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Parallel RF2 Execution                          step_id = 1650
global_flag = 0

sequence_no = 4 step_level = 2 parent_step_id = 1605 enabled_flag
= -1

iterator_name =                                             degree_parallelism
%DELETE_PARALLELISM%
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 = Throughput - Init RF1 Input          step_id = 1651
global_flag = 0

sequence_no = 1 step_level = 3 parent_step_id = 1632 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 = Throughput - RF1 - Stream%STREAM_NUM% step_id = 1652
global_flag = 0

sequence_no = 2 step_level = 3 parent_step_id = 1632 enabled_flag
= -1

iterator_name = degree_parallelism
%INSERT_PARALLELISM%
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 = Throughput - Init RF2 Input          step_id = 1653
global_flag = 0

sequence_no = 3 step_level = 3 parent_step_id = 1632 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 = Throughput - RF2 - Stream%STREAM_NUM% step_id = 1654
global_flag = 0

sequence_no = 4 step_level = 3 parent_step_id = 1632 enabled_flag
= -1

iterator_name = degree_parallelism
%DELETE_PARALLELISM%
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* = Throughput - Stream - Increment Update Set *step\_id* = 1655  
*global\_flag* = 0  
*sequence\_no* = 5 *step\_level* = 3 *parent\_step\_id* = 1632 *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* = Throughput\_RF\_Connection  
*parent\_version\_no* = 0.0  
*step\_text* = UPDATE TPCH\_AUX\_TABLE SET updateset=updateset+1

*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1633 *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 S.1

*step\_label* = Throughput - Query Stream 1 *step\_id* = 1656  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1633 *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* = 0.0  
*start\_directory* = Throughput\_Static\_Stream\_1\_Connection  
*parent\_version\_no* = 0.0  
*step\_text* =

*step\_label* = Throughput - Post to Semaphore (S1) *step\_id* = 1657  
*global\_flag* = 0

*step\_label* = Throughput - Query Stream 2 *step\_id* = 1658  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1634 *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* = 0.0  
*start\_directory* = Throughput\_Static\_Stream\_2\_Connection  
*parent\_version\_no* = 0.0  
*step\_text* =

*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1635 *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* = 0.0  
*start\_directory* = Throughput\_Static\_Stream\_3\_Connection  
*parent\_version\_no* = 0.0  
*step\_text* =

*step\_label* = Throughput - Post to Semaphore (S2) *step\_id* = 1659  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1634 *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 S.2

*step\_label* = Throughput - Query Stream 3 *step\_id* = 1660  
*global\_flag* = 0



*step\_label* = Throughput - Post to Semaphore (S3) *step\_id* = 1661  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1635 *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 S.3

*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1636 *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 S.4

*step\_label* = Throughput - Query Stream 4 *step\_id* = 1662  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1636 *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* = 0.0  
*start\_directory* = Throughput\_Static\_Stream\_4\_Connection  
*parent\_version\_no* = 0.0  
*step\_text* =

*step\_label* = Throughput - Post to Semaphore (S4) *step\_id* = 1663  
*global\_flag* = 0

```

step_label = Throughput - Query Stream 5                                step_id = 1664
global_flag = 0
sequence_no = 1 step_level = 3 parent_step_id = 1637 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 = 0.0
start_directory = Throughput_Static_Stream_5_Connection
parent_version_no = 0.0
step_text =

step_label = Throughput - Post to Semaphore (S5)                        step_id = 1665
global_flag = 0
sequence_no = 2 step_level = 3 parent_step_id = 1637 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 S.5

step_label = Throughput - Query Stream 6                                step_id = 1666
global_flag = 0
sequence_no = 1 step_level = 3 parent_step_id = 1638 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 = 0.0
start_directory = Throughput_Static_Stream_6_Connection
parent_version_no = 0.0
step_text =

```

*step\_label* = Throughput - Post to Semaphore (S6) *step\_id* = 1667  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1638 *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 S.6

*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1639 *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 S.7

*step\_label* = Throughput - Query Stream 7 *step\_id* = 1668  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1639 *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* = 0.0  
*start\_directory* = Throughput\_Static\_Stream\_7\_Connection  
*parent\_version\_no* = 0.0  
*step\_text* =

*step\_label* = Throughput - Post to Semaphore (S7) *step\_id* = 1669  
*global\_flag* = 0

*step\_label* = Throughput - Query Stream 8 *step\_id* = 1670  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1640 *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* = 0.0  
*step\_text* =

*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1641 *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* = 0.0  
*step\_text* =

*step\_label* = Throughput - Post to Semaphore (S8) *step\_id* = 1671  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1640 *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 S.8

*step\_label* = Throughput - Query Stream 9 *step\_id* = 1672  
*global\_flag* = 0

*step\_label* = Throughput - Post to Semaphore (S9) *step\_id* = 1673  
*global\_flag* = 0  
*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1641 *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 S.9

*sequence\_no* = 2 *step\_level* = 3 *parent\_step\_id* = 1642 *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 S.10

*step\_label* = Throughput - Query Stream 10 *step\_id* = 1674  
*global\_flag* = 0  
*sequence\_no* = 1 *step\_level* = 3 *parent\_step\_id* = 1642 *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* = 0.0  
*step\_text* =

*step\_label* = Throughput - Post to Semaphore (S10) *step\_id* = 1675  
*global\_flag* = 0

```

step_label = Execute RF1                                step_id = 1676
global_flag = 0

sequence_no = 1 step_level = 3 parent_step_id = 1646 enabled_flag
= -1

iterator_name = INSERT_SEGMENT 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 = --
-- Execute the Refresh RF1 Stored Procedure
--

EXEC RF1 %INSERT_SEGMENT%, %BATCH_SIZE%, %INSERT_PARALLELISM%,
%INSERT_EXECUTIONS%

```

```

parent_version_no = 0.0
step_text =

step_label = Load RF2 Input                                step_id = 1678
global_flag = 0

sequence_no = 1 step_level = 3 parent_step_id = 1648 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 = 0.0

start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0

step_text =

```

```

step_label = Load RF1 Input                                step_id = 1677
global_flag = 0

sequence_no = 1 step_level = 3 parent_step_id = 1644 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 = 0.0

start_directory = Dynamic_Connection_to_DB

```

```

--
EXEC RF1 %INSERT_SEGMENT%, %BATCH_SIZE%, %INSERT_PARALLELISM%,
%INSERT_EXECUTIONS%

step_label = Execute RF2                step_id = 1679
global_flag = 0

sequence_no = 1 step_level = 3          parent_step_id = 1650 enabled_flag
= -1

iterator_name = DELETE_SEGMENT          degree_parallelism 1
execution_mechanism = 1                 continuation_criteria = 2
failure_details =

step_file_name =
version_no = 0.0

start_directory = Power_Dynamic_DB_Connection
parent_version_no = 0.0

step_text = --
-- Execute the Refresh RF2 Stored Procedure
--

EXEC RF2 %DELETE_SEGMENT%, %BATCH_SIZE%, %DELETE_PARALLELISM%,
%DELETE_EXECUTIONS%

step_label = Throughput - Execute Stream%STREAM_NUM% RF1    step_id = 1680
global_flag = 0

sequence_no = 1 step_level = 4          parent_step_id = 1652 enabled_flag
= -1

iterator_name = INSERT_SEGMENT          degree_parallelism 1
execution_mechanism = 1                 continuation_criteria = 2
failure_details =

step_file_name =
version_no = 0.0

start_directory = Throughput_RF_Connection
parent_version_no = 0.0

step_text = --
-- Execute the Refresh RF1 Stored Procedure

step_label = Throughput - Execute Stream%STREAM_NUM% RF2    step_id = 1681
global_flag = 0

sequence_no = 1 step_level = 4          parent_step_id = 1654 enabled_flag
= -1

iterator_name = DELETE_SEGMENT          degree_parallelism 1
execution_mechanism = 1                 continuation_criteria = 2
failure_details =

step_file_name =
version_no = 0.0

start_directory = Throughput_RF_Connection
parent_version_no = 0.0

step_text = --
-- Execute the Refresh RF2 Stored Procedure
--

EXEC RF2 %DELETE_SEGMENT%, %BATCH_SIZE%, %DELETE_PARALLELISM%,
%DELETE_EXECUTIONS%

```

<p><i>step_label</i> = Throughput - Create RF1 Tables <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 1 <i>step_level</i> = 4 = -1</p> <p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %SETUP_DIR%\RF_Procs\RF1_init.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Dynamic_Connection_to_DB <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>step_id</i> = 1682</p> <p><i>parent_step_id</i> = 1651 <i>enabled_flag</i></p> <p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 1</p>	<p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %SETUP_DIR%\RF_Procs\RF1_index.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Dynamic_Connection_to_DB <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>
<p><i>step_label</i> = Throughput - Parallel Load RF1 Input <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 2 <i>step_level</i> = 4 = -1</p> <p><i>iterator_name</i> = %FILES_PER_UPDATE_SET%</p> <p><i>execution_mechanism</i> = 0 <i>failure_details</i> =</p> <p><i>step_file_name</i> = <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>step_id</i> = 1683</p> <p><i>parent_step_id</i> = 1651 <i>enabled_flag</i></p> <p><i>degree_parallelism</i> <i>continuation_criteria</i> = 0</p>	<p><i>iterator_name</i> =</p> <p><i>execution_mechanism</i> = 1 <i>failure_details</i> =</p> <p><i>step_file_name</i> = %SETUP_DIR%\RF_Procs\RF2_init.sql <i>version_no</i> = 0.0</p> <p><i>start_directory</i> = Dynamic_Connection_to_DB <i>parent_version_no</i> = 0.0</p> <p><i>step_text</i> =</p>	<p><i>degree_parallelism</i> 1 <i>continuation_criteria</i> = 2</p>
<p><i>step_label</i> = Throughput - Create RF1 Indices <i>global_flag</i> = 0</p> <p><i>sequence_no</i> = 3 <i>step_level</i> = 4 = -1</p>	<p><i>step_id</i> = 1684</p> <p><i>parent_step_id</i> = 1651 <i>enabled_flag</i></p>		



```

step_label = Throughput - Parallel Load RF2 Input          step_id = 1686
global_flag = 0
sequence_no = 2 step_level = 4 parent_step_id = 1653 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 = 0.0
step_text =

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 = 0.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Throughput - Load RF2 Input          step_id = 1689
global_flag = 0
sequence_no = 1 step_level = 5 parent_step_id = 1686 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 = 0.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Throughput - Create RF2 Indices          step_id = 1687
global_flag = 0
sequence_no = 3 step_level = 4 parent_step_id = 1653 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 = 0.0
start_directory = Dynamic_Connection_to_DB
parent_version_no = 0.0
step_text =

step_label = Throughput - Load RF1 Input          step_id = 1688
global_flag = 0
sequence_no = 1 step_level = 5 parent_step_id = 1683 enabled_flag
= -1

```

15	1662	0.0	1 1
15	1660	0.0	1 1
15	1660	0.0	3 1
15	1660	0.0	2 22
15	1658	0.0	2 22
15	1658	0.0	1 1
15	1658	0.0	3 1
15	1656	0.0	1 1

## *Iterator\_values*

<i>workspace_id</i>	<i>step_id</i>	<i>version_no</i>	<i>type</i>	<i>iterator_value</i>
15	1662	0.0	2	22
15	1670	0.0	1	1
15	1670	0.0	2	22
15	1670	0.0	3	1
15	1668	0.0	2	22
15	1668	0.0	1	1
15	1668	0.0	3	1
15	1666	0.0	1	1
15	1666	0.0	2	22
15	1666	0.0	3	1
15	1664	0.0	3	1
15	1664	0.0	2	22
15	1632	0.0	3	1
15	1662	0.0	3	1
15	1672	0.0	3	1

*sequence\_no*

0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0
0

15	1656	0.0	2 22	015	1677	0.0	2 %FILES_PER_UPDATE_SET%
15	1656	0.0	3 1	015	1677	0.0	1 1
15	1632	0.0	2 %MAX_STREAMS%	015	1676	0.0	2 %INSERT_EXECUTIONS%
15	1632	0.0	1 1	015	1676	0.0	3 1
15	1664	0.0	1 1	015	1676	0.0	1 1
15	1678	0.0	3 1	015	1674	0.0	2 22
15	1689	0.0	1 1	015	1674	0.0	1 1
15	1689	0.0	3 1	0			
15	1688	0.0	3 1	0			
15	1688	0.0	2 %FILES_PER_UPDATE_SET%	0			
15	1688	0.0	1 1	0			
15	1681	0.0	1 1	0			
15	1681	0.0	2 %DELETE_EXECUTIONS%	0			
15	1681	0.0	3 1	0			
15	1680	0.0	3 1	0			
15	1680	0.0	2 %INSERT_EXECUTIONS%	0			
15	1680	0.0	1 1	0			
15	1679	0.0	1 1	0			
15	1672	0.0	1 1	0			
15	1679	0.0	3 1	0			
15	1672	0.0	2 22	0			
15	1678	0.0	1 1	0			
15	1678	0.0	2 %FILES_PER_UPDATE_SET%	0			
15	1677	0.0	3 1	0			

15	1674	0.0	3 1	0
15	1689	0.0	2 %FILES_PER_UPDATE_SET%	0
15	1679	0.0	2 %DELETE_EXECUTIONS%	0

## **APPENDIX G : DISK CONFIGURATION**

The row partitions used to hold data, indexes, database log and temporary database (TempDB).

The raw partitions for the database and the NTFS partition for the backup and flat files were mounted to a folder on the E: drive below.

E:\mnt\li  
E:\mnt\ge  
E:\mnt\idx  
E:\mnt\load  
E:\mnt\tempdb  
E:\mnt\backup  
E:\mnt\ffs  
E:\mnt\ACID

A description of distribution of database filegroups and log can be found in the table below.

FC HBA Number	Disk Subsystem					Logical Unit					
	System Number	Controller Number	port	RAID Level	Number of Drives						
0	0	0	A	1	2	Root (29GB)					
0	0	0	A	1	2	E: drive (29GB)					
0	0	0	A	0+1	10	log (147GB)					
0	0	0	A	5	7	Backup (175GB)					
0	0	0	A	5	7	Backup (175GB)					
1	0	0	B	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
8	0	0	C	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
9	0	0	D	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
2	0	1	A	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
3	0	1	B	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	
10	0	1	C	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	
11	0	1	D	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	
0	0	0	A	0	2	ACID (66GB)					
0	0	0	A	0	2	ACID (66GB)					
4	1	0	A	5	7	Backup (175GB)					
4	1	0	A	5	7	Backup (198GB)					
4	1	0	A	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
5	1	0	B	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
12	1	0	C	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
13	1	0	D	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
6	1	1	A	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	
7	1	1	B	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	
14	1	1	C	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	
15	1	1	D	0	16	LINEITEM_FG (80GB)	GENERAL_F G	INDEX_F G	LOAD_FG (80GB)	TempDB (80GB)	Flatfiles (100GB)
4	0	1	A	0+1	4	ACID (66GB)					

**APPENDIX H : PRICE QUOTATION**



Hitachi America, Limited / Server Systems Group  
 2000 Sierra Point Parkway 3/3  
 Brisbane, CA 94005-1835  
 Tel:1-866-Hitachi (448-2244)

**Date:** 4/17/2006  
**To:** Hitachi, Ltd. Enterprise Server Division  
 1Horiyamashita Hadano-shi Kanagawa-ken, 259-1392 Japan  
**Attn:** Mt. Hiromichi Shibata

It is our pleasure to present you with the following quotation for BladeSymphony products for your TPC-H testing. All prices showing below are US Dollars.

#	Category	Model Name	Specifications	Model Number	List Price	Qty	Extended List Price	Proposed Price
1	Chassis	Chassis BS1000B (A2)	Chassis Type B	GVX-SREA2B51N	\$6,545	1	\$6,545	
2		Power Supply Module	200V/1500W	GVX-BP9200VBX	\$265	2	\$530	
3		I/O Module TYPE-1	TYPE1/max 8PCI slot, 2slot/Server Mod	GVX-BE9SLT1BX	\$1,558	2	\$3,116	
4		Dummy Server Module		GVX-BE9BLDMYBX	\$39	3	\$117	
5		Dummy SVP Module		GVX-BE9SVPDMYBX	\$82	1	\$82	
6	Server Module	Xeon Server Module	Xeon 3.D GHz/1MB x1	GVAX51A2-1NN0N4X	\$2,407	1	\$2,407	
7		IPF 2-way Server Module	Itanium2 1.66GHz/9MB x2, FSB667	GVAX75A2-7NN0N5X	\$16,147	4	\$64,588	
8		IPF SMP Upgrade License (2way to 4way) (Itanium2 1.66GHz/9MB)			\$1,808	1	\$1,808	
9		IPF SMP Upgrade License (4way to 8way) (Itanium2 1.66GHz/9MB)			\$155,676	1	\$155,676	
10	Memory	1 GB Memory (DDR1)	1GB(DDR1_266_512MBx2) for Xeon	GVX-MJ701GTEX	\$277	1	\$277	
11		8 GB Memory (DDR2)	8GB(DDR2_400_2048MBx4) for IPF	GVX-MJ908GAEX	\$3,790	16	\$60,640	
12	HDD	HDD 160 GB	160GB, 7200 min-1	GVX-UH7160NEX	\$129	2	\$258	
13	PCI Adapter	4-Gbps FC Adapter (2 Ch)	4Gb,2ch,PCI-X	GVX-CC64G2BX	\$590	16	\$9,440	
14	Rack Cabinet	16 U Rack Cabinet Set	16 U Rack Cabinet Set	GVX-SRK7165BK	\$1,046	1	\$1,046	
15		16 U Front Door BY	16U(Black x Yellow)	GVX-RD9166BK	\$384	1	\$384	
16		16 U Rack Side Panel	16U(Black):A couple of two pieces	GVX-RS7165BK	\$147	1	\$147	
17		Blank Panel	for 16U Rack /38U Rack, 1U(Black)	GVX-RP7011BK	\$6	2	\$12	
18	PDU	Outlet Box 3N	3 outlet (200V) for North America	GVX-AG9203A	\$675	1	\$675	
19	Cable	KVM Cable	for Keyboard, Mouse, Display, 1m	GVX-LY7201	\$70	1	\$70	
20	Console	17" SXGA Console Drawer FD-1000 series	w/o KVM	NC14010-B452/US	\$2,493	1	\$2,493	
21		KVM Cascade Cable (USB)	1.8m	NC14000-B102	\$35	1	\$35	
22		Guide Rail		NC14003-T591	\$52	1	\$52	
23	FD drive	Floppy Drive External USB Floppy Drive		DC361B	\$73	1	\$73	
24	CD drive	24X External DVD/CDRW USB2.0		EZ-WTR-DVD-C24	\$158	1	\$158	
<b>Hardware Sub Total</b>						<b>63</b>	<b>\$310,629</b>	<b>\$230,333</b>
25	Maintenance	Express	3 year maintenance service			1	\$210,733	\$91,558
<b>Total</b>							<b>\$521,362</b>	<b>\$321,891</b>





Date: 4/17/2006  
 To: Hitachi, Ltd. Enterprise Server Division  
 1 Horiyamashita, Hadano-shi, Kanagawa-ken, 259-1392 Japan  
 Attn: Mr. Hiromichi Shibata

Hitachi Data Systems  
 750 Central Expressway,  
 Santa Clara, CA 95050-2627  
 408-970-1000

Please find following price quotation for the reference for TPC-H testing.

\$US

No.	System#	Category	Product Code	Product Description	List Price	Proposed Price	Q'ty	Extended ListPrice	Extended Proposed Price	
1	System_1	Hardware	DF600-RKA	9500V Storage Expansion Unit (RKA)	6,300.00	3,465.00	11	69,300.00	38,115.00	
2			DF600-RKHE	9585V Base Unit(RKHE) w/2 controllers 4 FC ports	38,000.00	20,900.00	1	38,000.00	20,900.00	
3			DF-F600-AEH72	9500V 72GB FC 15K RPM HDD	1,410.00	776.00	164	231,240.00	127,264.00	
4			DF-F600-C1GX2H	9580V 2GB Cache Memory Module 2x1024MB	4,830.00	2,656.00	4	19,320.00	10,624.00	
5			DF-F600-DH2G2	9580V Fibre Channel Intfc 2Gbps w/2 FC ports	3,800.00	2,090.00	2	7,600.00	4,180.00	
6			JZ-050LL005PC.P	LC/LC Duplex FibreOptic Plenum Cable 5M	185.00	114.70	8	1,480.00	917.60	
								366,940.00	202,000.60	
7		Software	044-210195-03.P	9570/85V Resource Manager License	3,500.00	1,925.00	1	3,500.00	1,925.00	
								3,500.00	1,925.00	
8	System_2	Hardware	DF600-RKA	9500V Storage Expansion Unit (RKA)	6,300.00	3,465.00	11	69,300.00	38,115.00	
9			DF600-RKHE	9585V Base Unit(RKHE) w/2 controllers 4 FC ports	38,000.00	20,900.00	1	38,000.00	20,900.00	
10			DF-F600-AEH72	9500V 72GB FC 15K RPM HDD	1,410.00	776.00	164	231,240.00	127,264.00	
11			DF-F600-C1GX2H	9580V 2GB Cache Memory Module 2x1024MB	4,830.00	2,656.00	4	19,320.00	10,624.00	
12			DF-F600-DH2G2	9580V Fibre Channel Intfc 2Gbps w/2 FC ports	3,800.00	2,090.00	2	7,600.00	4,180.00	
13			JZ-050LL005PC.P	LC/LC Duplex FibreOptic Plenum Cable 5M	185.00	114.70	8	1,480.00	917.60	
								366,940.00	202,000.60	
14		Software	044-210195-03.P	9570/85V Resource Manager License	3,500.00	1,925.00	1	3,500.00	1,925.00	
								3,500.00	1,925.00	
15	Others	Hardware	DF-F600-J1H	9500V 125V AC 15A Power Cord	39.00	21.00	48	1,872.00	1,008.00	
								1,872.00	1,008.00	
16		Rack		122296-1-P.P	Filler Panel 1.0U perf. for HDS Rack	37.00	20.00	4	148.00	80.00
17				7846400.P	19inch Rack for HDS-US	4,924.00	2,708.00	2	9,848.00	5,416.00
18				7846406.P	Universal Rails (Kit)	84.00	46.00	24	2,016.00	1,104.00
									12,012.00	6,600.00
19		Installation		-	Installation	2,350.00	2,350.00	1	2,350.00	2,350.00
								2,350.00	2,350.00	
20	Support Fee		-	Support Fee (Standard support of 9570/85V Resource Manager, Annual)	43.75	43.75	6	262.50	262.50	
21			-	Support Fee (Standard support of hardware; Annual)	9,943.00	9,943.00	4	39,772.00	39,772.00	
22			-	Support Fee (Standard support of hardware Upgrade Yr 1)	6,615.00	6,615.00	2	13,230.00	13,230.00	
								53,264.50	53,264.50	
<b>Total</b>								<b>810,378.50</b>	<b>471,073.70</b>	

Microsoft Corporation  
One Microsoft Way  
Redmond, WA 98052-6399

Tel 425 882 8080  
Fax 425 936 7329  
<http://www.microsoft.com/>

**Microsoft**

April 25, 2006

Hitachi, Ltd.  
Tsuyoshi Tanaka  
1-280, Higashi-  
koigakubo  
Kokubunji-shi  
Tokyo, 185-8601

Tanaka-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	<b>SQL Server 2005 Enterprise Itanium Edition with Service Pack 1</b> <i>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.</i>	\$8,487	1	\$8,487
359-01911	<b>SQL Server 2005 Client License</b> <i>Discount Schedule: Open Program - No Level Unit Price reflects a 4% discount from the retail unit price of \$163.</i>	\$156	35	\$5,460
P72-00284	<b>Windows Server 2003 Enterprise Edition For 64-bit Itanium-based Systems</b> <i>Server License Only - No CALs Discount Schedule: Open Program - No Level Unit Price reflects a 42% discount from the retail unit price of \$3,999.</i>	\$2,334	1	\$2,334
P73-00295	<b>Windows Server 2003 Standard Edition</b> <i>Server License Only - No CALs Discount Schedule: Open Program - No Level Unit Price reflects a 28% discount from the retail unit price of \$999.</i>	\$719	1	\$719
254-00170	<b>Visual C++ Standard Edition</b> <i>No Discounts Applied</i>	\$109	1	\$109
046-00856	<b>Visual Basic 2003 .Net Professional</b> <i>No Discounts Applied</i>	\$109	1	\$109
N/A	<b>Microsoft Problem Resolution Services</b> <i>Professional Support</i>	\$245	1	\$245

	(1 Incident)			
--	--------------	--	--	--

All products are currently orderable through Microsoft's normal distribution.

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](mailto:jamiere@microsoft.com).

Reference ID: PHTsta0603020079.

Please include this Reference ID in any correspondence regarding this price quote.