TPC Benchmark™ H Full Disclosure Report

for

IBM® System x™ 3950 M2

using

Microsoft® SQL Server 2005

Submitted for Review

November 28, 2007

Amended: February 18, 2008

First Edition - November 2007

THE INFORMATION CONTAINED IN THIS DOCUMENT IS DISTRIBUTED ON AN AS IS BASIS WITHOUT ANY WARRANTY EITHER EXPRESSED OR IMPLIED. The use of this information or the implementation of any of these techniques is the customer's responsibility and depends on the customer's ability to evaluate and integrate them into the customer's operational environment. While each item has been reviewed by IBM for accuracy in a specific situation, there is no guarantee that the same or similar results will be obtained elsewhere. Customers attempting to adapt these techniques to their own environment do so at their own risk.

In this document, any references made to an IBM licensed program are not intended to state or imply that only IBM's licensed program may be used; any functionally equivalent program may be used.

This publication was produced in the United States. IBM may not offer the products, services, or features discussed in this document in other countries, and the information is subject to change without notice. Consult your local IBM representative for information on products and services available in your area.

© Copyright International Business Machines Corporation 2007. All rights reserved.

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

U.S. Government Users - Documentation related to restricted rights: Use, duplication, or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract with IBM Corp.

Trademarks

IBM, the IBM logo, System x and System Storage are trademarks or registered trademarks of International Business Machines Corporation.

The following terms used in this publication are trademarks of other companies as follows: TPC Benchmark, TPC-H, QppH QthH and QphH are trademarks of Transaction Processing Performance Council; Intel and Xeon are trademarks or registered trademarks of Intel Corporation; Microsoft and Windows are registered trademarks of Microsoft Corporation in the United States and/or other countries, or both. Other company, product, or service names, which may be denoted by two asterisks (**), may be trademarks or service marks of others.

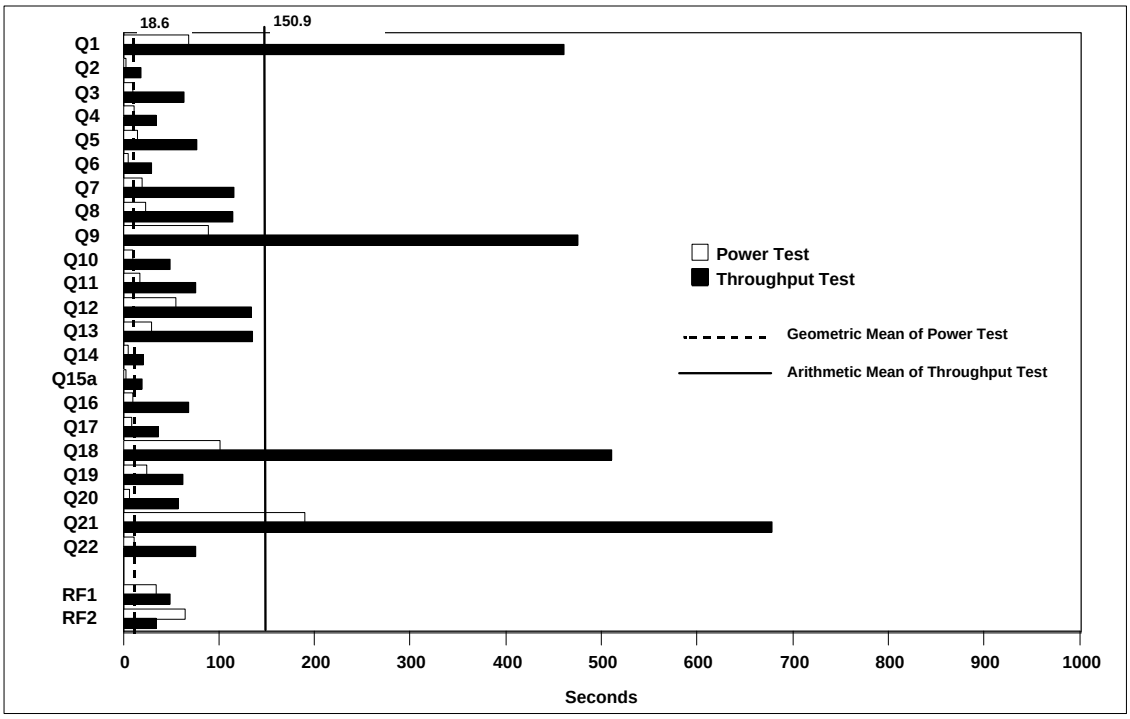
Notes

[1] GHz only measures microprocessor internal clock speed, not application performance. Many factors affect application performance.
[2] When referring to hard disk capacity, one GB equals one billion bytes. Total user-accessible capacity may be less.

IBM Corporation	IBM® System x™ 3950 M2 with Microsoft® SQL Server 2005	TPC-H Rev 2.6.1 Report Date: Nov. 28, 2007 Amended: Feb. 18, 2008

Total System Cost	Composite Query-per-Hour Metric	Price/Performance
$ 248,635 USD	**46,034.4** **QphH @ 300GB**	**$ 5.40 USD** **per QphH @ 300GB**

Database Size	Database Manager	Operating System	Other Software	Availability Date
300GB	**Microsoft SQL Server 2005 Enterprise x64 Editon**	**Microsoft Windows® Server 2003 R2 Enterprise x64 Edition SP2**	**Microsoft Visual Studio Standard 2005**	**Mar. 7, 2008**



Database Load Time: 7:21:03	Load Included Backup: Y	Total Data Storage / Database Size: 74.38
RAID (Base Tables Only): N	**RAID (Base Tables and Auxiliary Data Structures): N**	**RAID (All): N**

Configuration		
Processors/Cores/Threads	**8/32/32**	**Intel Xeon Processor X7350 (2.93GHz, 4MB L2 Cache per 2 Cores)**
Memory	**32**	**8GB (2 x 4GB) PC2-5300 ECC DDR2 SDRAM RDIMM**
Disk Controllers	**13**	**IBM ServeRAID-MR10 SAS/SATA Controller**
Disk Drives	**300**	**73.4GB 15K SAS Drive**
	4	**73.4GB 10K SFF SAS Drive**
Total Disk Storage		**22314GB**

IBM Corporation	IBM System x3950 M2 with Microsoft SQL Server 2005	TPC-H Rev 2.6.1 Report Date: Nov. 28, 2007 Amended: Feb. 18, 2008

Description	Part Number	Brand	Price Source	Unit Price	Quantity	Extended Price	3-Yr. Maint. Price
Server Hardware							
IBM System x3950 M2 (2 x Intel Xeon Processor X7350 with 2.93GHz/2x4MB L2 Cache, 4 Memory Cards, 8 x 1GB DIMM, onboard SAS enablement key, cables, integrated dual-port 10/100/1000 Gigabit Ethernet)	7141-4SU	IBM	1b	20,499	2	40,998	
Intel Xeon Processor 7350 (2.93GHz/1066MHz FSB/2x4MB L2)	44E4243	IBM	1b	3,579	4	14,316	
8GB (2x4GB) 667MHz PC2-5300 ECC DDR2 SDRAM DIMM	41Y2768	IBM	1b	1,699	32	54,368	
73GB 10K 2.5" Hot-Swap SAS SFF	40K1052	IBM	1b	319	4	1,276	
IBM ServeRAID-MR10M SAS/SATA Controller	43W4339	IBM	1b	1,049	13	13,637	
IBM SAS 3m Cable	39R6531	IBM	1b	135	25	3,375	
IBM T115 15-inch TFT Display	494215U	IBM	1a	209	1	209	
IBM Preferred Pro USB Keyboard	40K9584	IBM	1b	29	1	29	
IBM 3-Button Optical Mouse - Black - USB	40K9201	IBM	1b	19	1	19	
ServicePac for 3-Year 24x7x4 Support (x3950 M2)	96P2688	IBM	1b	3,390	2		6,780
ServicePac for 3-Year 24x7x4 Support (Display)	10N3110	IBM	1b	90	1		90
					Subtotal	128,227	6,870
Server Storage							
IBM System Storage EXP3000	172701X	IBM	1b	3,199	25	79,975	
IBM 73.4GB 15K SAS HDD	40K1043	IBM	1b	309	300	92,700	
IBM UPS 750TLV	2130R1X	IBM	1b	299	1	299	
IBM S2 42U Standard Rack	93074RX	IBM	1b	1,489	2	2,978	
ServicePac for 3-Year 24x7x4 Support (EXP3000)	41L2768	IBM	1b	760	25		19,000
ServicePac for 3-Year 24x7x4 Support (Rack)	41L2760	IBM	1b	300	2		600
					Subtotal	175,952	19,600
Server Software							
Microsoft SQL Server 2005 Enterprise x64 Edition	810-04779	Microsoft	2a	13,969	1	13,969	
Microsoft SQL Server 2005 Client License	359-01911	Microsoft	2a	163	35	5,705	
Microsoft Windows Server 2003 R2 Enterprise x64 Edition	P72-01684	Microsoft	2a	3,999	1	3,999	
Microsoft Visual Studio Standard 2005	127-00012	Microsoft	2b	250	1	250	
Microsoft Problem Resolution Services		Microsoft	2b	245	1		245
					Subtotal	23,923	245
					Total	328,102	26,715
IBM Large Purchase Discount (See Note 1.)	29.85%		1b			98,621	
Microsoft Open Program Discount Schedule	31.94%		2a			7,561	

Pricing: 1 - IBM - 1-888-SHOP-IBM, ext. 5821; 2 - Microsoft
Note 1: Discount based on IBM Direct guidance applies to all line items where Pricing=1b.
Pricing is for this system or one of similar size.

Three-Year Cost of Ownership USD:	$248,635
QphH@300GB:	46,034.4
$ USD/QphH@300GB:	$5.40

Audited by Francois Raab, InfoSizing, Inc. (www.sizing.com)

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

Measurement Results:

Database Scale Factor	300
Total Data Storage/Database Size	74.38
Start of Database Load	11-22-07 8:20:33
End of Database Load	11-22-07 15:41:36
Database Load Time	7:21:03
Query Streams for Throughput Test	6
TPC-H Power	58,196.9
TPC-H Throughput	36,413.8
TPC-H Composite Query-per-Hour (QphH@300GB)	46,034.4
Total System Price over 3 Years	$ 248,635 USD
TPC-H Price/Performance Metric ($/QphH@300GB)	$ 5.40 USD

Measurement Interval:

Measurement Interval in Throughput Test (Ts) = 3,915

Duration of Stream Execution:

	Seed	Query Start Date/Time Query End Date/Time	RF1 Start Date/Time RF1 End Date/Time	RF2 Start Date/Time RF2 End Date/Time	Duration
Stream 0	1122154136	11/22/07 17:01:56 11/22/07 17:15:32	11/22/07 17:01:56 11/22/07 17:02:52	11/22/07 17:14:56 11/22/07 17:15:31	00:13:36
Stream 1	1122154137	11/22/07 17:15:32 11/22/07 18:10:50	11/22/07 18:12:25 11/22/07 18:13:19	11/22/07 18:13:19 11/22/07 18:13:55	00:55:18
Stream 2	1122154138	11/22/07 17:15:32 11/22/07 18:12:11	11/22/07 18:13:55 11/22/07 18:14:45	11/22/07 18:14:45 11/22/07 18:15:19	00:56:39
Stream 3	1122154139	11/22/07 17:15:32 11/22/07 18:08:33	11/22/07 18:15:19 11/22/07 18:16:09	11/22/07 18:16:09 11/22/07 18:16:42	00:53:00
Stream 4	1122154140	11/22/07 17:15:32 11/22/07 18:11:27	11/22/07 18:16:42 11/22/07 18:17:27	11/22/07 18:17:27 11/22/07 18:18:00	00:55:54
Stream 5	1122154141	11/22/07 17:15:32 11/22/07 18:09:47	11/22/07 18:18:00 11/22/07 18:18:47	11/22/07 18:18:47 11/22/07 18:19:23	00:54:14
Stream 6	1122154142	11/22/07 17:15:32 11/22/07 18:12:25	11/22/07 18:19:23 11/22/07 18:20:08	11/22/07 18:20:08 11/22/07 18:20:47	00:56:52

IBM Corporation	IBM System x3950 M2 with Microsoft SQL Server 2005	TPC-H Rev 2.6.1 Report Date: Nov. 28, 2007 Amended: Feb. 18, 2008

TPC-H Timing Intervals (in seconds):

Query	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
Stream 0	68.5	2.5	10.2	11.7	15.7	5.4	19.9	24.1	89.0	10.5	17.1	54.7
Stream 1	481.3	18.0	60.0	28.3	77.1	24.0	110.4	108.7	415.1	38.2	86.9	161.6
Stream 2	483.4	23.1	61.0	34.1	72.9	29.6	180.6	110.3	493.5	60.2	77.2	175.2
Stream 3	482.0	18.8	72.7	41.1	85.6	34.3	125.0	131.7	438.4	41.4	60.9	127.6
Stream 4	415.0	16.6	44.8	37.5	85.4	33.4	111.2	102.9	600.1	41.5	85.8	112.4
Stream 5	434.1	21.3	70.9	31.3	73.7	21.0	82.7	97.0	462.9	35.6	67.4	95.2
Stream 6	467.7	18.2	74.1	36.2	69.5	37.8	87.9	136.8	442.0	79.3	79.6	136.0
Minimum	415.0	16.6	44.8	28.3	69.5	21.0	82.7	97.0	415.1	35.6	60.9	95.2
Average	460.6	19.3	63.9	34.5	77.4	30.0	116.3	114.6	475.3	49.4	76.3	134.7
Maximum	483.4	23.1	74.1	41.1	85.6	37.8	180.6	136.8	600.1	79.3	86.9	175.2

Stream ID	Q13	Q14	Q15a	Q16	Q17	Q18	Q19	Q20	Q21	Q22	RF1	RF2
Stream 0	29.9	5.0	3.2	10.8	9.7	101.4	25.2	6.5	190.5	12.1	56.9	35.1
Stream 1	111.7	18.0	20.0	66.0	36.5	497.2	46.0	66.8	783.5	62.2	54.5	35.5
Stream 2	142.8	31.5	19.0	75.1	58.3	493.4	72.5	47.0	570.7	87.6	50.2	34.1
Stream 3	108.9	19.2	18.8	65.0	40.6	443.7	72.2	56.1	609.5	87.1	49.8	32.7
Stream 4	144.7	18.5	22.9	76.7	33.0	469.6	54.8	35.0	756.9	57.8	45.0	32.8
Stream 5	142.3	22.9	21.6	70.4	28.8	451.7	66.1	94.7	782.8	79.9	47.2	35.5
Stream 6	160.3	20.2	20.9	57.8	21.7	706.6	65.6	43.9	569.9	80.2	45.6	38.3
Minimum	108.8	18.0	18.8	57.8	21.7	443.7	46.0	35.0	569.9	57.8	45.0	32.7
Average	135.1	21.7	20.5	68.5	36.5	510.4	62.9	57.3	678.9	75.8	48.7	34.8
Maximum	160.3	31.5	22.9	76.7	58.3	706.6	72.5	94.7	783.5	87.6	54.5	38.3

Table of Contents

Preface 12
1 General Items 14
1.1 Benchmark Sponsor 14
1.2 Parameter Settings 14
1.3 Configuration Diagrams 14
1.3.1 Priced and Measured Configurations 15
2 Clause 1: Logical Database Design Related Items 16
2.1 Database Table Definitions 16
2.2 Database Organization 16
2.3 Horizontal Partitioning 16
2.4 Replication 16
3 Clause 2: Queries and Update Functions Related Items 17
3.1 Query Language 17
3.2 Random Number Generation 17
3.3 Substitution Parameters Generation 17
3.4 Query Text and Output Data from Database 17
3.5 Query Substitution Parameters and Seeds Used 17
3.6 Query Isolation Level 17
3.7 Refresh Function Implementation 18
4 Clause 3: Database System Properties Related Items 19
4.1 Atomicity Requirements 19
4.1.1 Atomicity of Completed Transactions 19
4.1.2 Atomicity of Aborted Transactions 19
4.2 Consistency Requirements 19
4.2.1 Consistency Tests 20
4.3 Isolation Requirements 20
4.3.1 Isolation Test 1 - Read-Write Conflict with Commit 20
4.3.2 Isolation Test 2 - Read-Write Conflict with Rollback 20
4.3.3 Isolation Test 3 - Write-Write Conflict with Commit 20
4.3.4 Isolation Test 4 - Write-Write Conflict with Rollback 21
4.3.5 Isolation Test 5 - Concurrent Read and Write Transactions on Different Tables 21
4.3.6 Isolation Test 6 - Update Transactions during Continuous Read-Only Query Stream 21
4.4 Durability Requirements 22
4.4.1 Loss of Log and Failure of a Durable Medium 22
4.4.2 System Crash 22
4.4.3 Memory Failure 22
5 Clause 4: Scaling and Database Population Related Items 23
5.1 Cardinality of Tables 23
5.2 Distribution of Tables and Logs 23
5.3 Database Partition / Replication Mapping 27
5.4 RAID Implementation 27
5.5 DBGEN Modifications 28
5.6 Database Load Time 28
5.7 Data Storage Ratio 28
5.8 Database Load Mechanism Details and Illustration 28
5.9 Qualification Database Configuration 29
6 Clause 5: Performance Metrics and Execution Rules Related Items 30
6.1 System Activity between Load and Performance Tests 30
6.2 Steps in the Power Test 30
6.3 Timing Intervals for Each Query and Refresh Function 30

6.4 Number of Streams for the Throughput Test . . . 30
6.5 Start and End Date/Times for Each Query Stream . . . 30
6.6 Total Elapsed Time for the Measurement Interval . . . 30
6.7 Refresh Function Start Date/Time and Finish Date/Time . . . 30
6.8 Timing Intervals for Each Query and Each Refresh Function for Each Stream . . . 31
6.9 Performance Metrics . . . 31
6.10 Performance Metric and Numerical Quantities from Both Runs . . . 31
6.11 System Activity between Tests . . . 31
7 Clause 6: SUT and Driver Implementation Related Items . . . 32
7.1 Driver . . . 32
7.2 Implementation-Specific Layer . . . 32
Power Run . . . 32
Throughput Run . . . 32
7.3 Profile-Directed Optimization . . . 32
8 Clause 7: Pricing Related Items . . . 33
8.1 Hardware and Software Components . . . 33
8.2 Three-Year Cost of System Configuration . . . 33
8.3 Availability Dates . . . 33
8.4 Country-Specific Pricing . . . 33
Clause 9: Audit Related Items . . . 34
9.1 Auditor's Report . . . 34
Appendix A: Tunable Parameters . . . 37
Microsoft SQL Server 2005 Version . . . 37
Microsoft SQL Server 2005 Startup Parameters . . . 37
Microsoft SQL Server 2005 Installation . . . 37
Microsoft SQL Server 2005 Configuration Parameters . . . 37
Microsoft Windows Server 2003 Configuration . . . 38
Windows Registry Modifications . . . 38
ServeRAID-MR10 Configuration . . . 39
Windows System Information . . . 81
Appendix B: Database Build Scripts . . . 119
BackupDatabase.sql . . . 119
BackupDatabase_2.sql . . . 119
CreateBackupDevices.sql . . . 119
CreateBackupDevices_2.sql . . . 119
CreateDatabase.sql . . . 119
CreateFK.sql . . . 121
CreateIndexes_1.sql . . . 121
CreateIndexes_2.sql . . . 122
CreateTables.sql . . . 122
DropLoadFG.sql . . . 123
RestoreDatabase.sql . . . 123
MoveTempDB.sql . . . 123
ResizeTempDB.sql . . . 123
Appendix C: Qualification Query Output . . . 124
Qualification Queries . . . 124
Query 1 . . . 124
Query 2 . . . 124
Query 3 . . . 124
Query 4 . . . 124
Query 5 . . . 124
Query 6 . . . 124
Query 7 . . . 125
Query 8 . . . 125

Query 9 125
Query 10 125
Query 11 126
Query 12 126
Query 13 126
Query 14 127
Query 15 127
Query 16 127
Query 17 127
Query 18 127
Query 19 127
Query 20 127
Query 21 127
Query 22 128
Query Substitution Parameters 128
Power Run Stream 0 128
Stream 1 128
Stream 2 129
Stream 3 129
Stream 4 129
Stream 5 129
Stream 6 129
Appendix D: Refresh Function Source Code 131
AddEnvVar.vbs 131
Call_Create_RFStreams 131
Clrfs_env 131
Create_RF1_Indices.sql 131
Create_RF1_Tables.sql 131
Create_RF2_Indices.sql 131
Create_RF2_Tables.sql 131
CreateRF2proc.sql 131
Create-RF1_Whole 133
Create-RF2_Whole 133
Create-RF_Streams 134
ExecSQLFileList 135
ExecSQLFileListSemaphore 135
Execute_RF1_tmp.sql 135
Execute_RF2_tmpl.sql 135
Increment_Update_Set.sql 135
installEnvVariables 135
Load_RF1_tmpl.sql 136
Load_RF2_tmpl.sql 136
precreatefiles_300G 136
RF1 Command Invocations 136
RF1_Whole 136
RF2_Whole 137
RFStreams_Run1 137
RFStreams_Run2 138
Run_rf1whole 138
WaitForSemaphore 138
Appendix E: Implementation Specific Layer and Source Code 139
cArrConstraints.cls 139
cArrParameters.cls 143
cArrSteps.cls 145

cAsyncShell.cls 147
cConnDtl.cls 148
cConnDtls.cls 151
cConnection.cls 152
cConnections.cls 156
cConstraint.cls 157
cFailedStep.cls 161
cFailedSteps.cls 161
cFileInfo.cls 162
cFileSM.cls 162
cGlobalStep.cls 163
cInstance.cls 168
cInstances.cls 172
cIterator.cls 173
cManager.cls 176
cNode.cls 180
cNodeCollections.cls 180
Common.bas 183
ConnDtlCommon.bas 188
ConnectionCommon.bas 188
cParameter.cls 189
cRunColIt.cls 192
cRunInst.cls 193
cRunItDetails.cls 240
cRunItNode.cls 241
cRunOnly.cls 241
cRunStep.cls 241
cRunWorkspace.cls 248
CSequence.cls 253
CStack.cls 254
CStep.cls 254
CStepTree.cls 264
CStringSM.cls 264
CSubStep.cls 266
CSubSteps.cls 268
CTermProcess.cls 268
CTermStep.cls 269
CTermSteps.cls 269
CTimerSM.csl 269
CVBErrorsSM.cls 270
CVector.cls 271
CVectorLng.cls 272
CVectorStr.cls 274
CWorker.cls 275
CWorkspace.cls 279
DatabaseSM.bas 282
DebugSM.bas 289
ErrHandling.bas 292
FileCommon.bas 292
IteratorCommon.bas 293
MsgConfirm.bas 295
OpenFiles.bas 295
ParameterCommon.bas 297
Public.bas 298

RunCommon.bas 299
RunInstHelper.bas 300
ShellSM.bas 301
SMErr.bas 301
SortSM.bas 304
Startup.bas 305
StepCommon.bas 306
TimerSM.bas 309
ToolsCommon.bas 309
TPCHReports.bas 311
Utility.bas 324
WindowsApiCommon.bas 326
WorkspaceCommon.bas 328
Appendix F: Price Quotations 332

Preface

TPC Benchmark H Standard Specification was developed by the Transaction Processing Performance Council (TPC). It was released on February 26, 1999, and most recently revised (Revision 2.6.1). This is the full diclosure report for benchmark testing of the IBM System x3850 M2 according to the TPC Ben5hmark H Standard Specification.

The TPC Benchmark H is a decision support benchmark. It consists of a suite of business-oriented ad hoc queries and concurrent data modifications. The queries and the data populating the database have been chosen to have broad industrywide 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 set 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 with 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 data modifications against all tables, exeat 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 ongoing refresh functions, which batch together a number of modifications impacting some part of the decision support database.
- Due to the worldwide nature of the business data stored in the TPC-H database, the queries and the refresh functions may be executed against the database at any time, especially in relation to each other. In addition, this mix of queries and refresh functions is subject to specific ACIDity requirements, since queries and refresh functions 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 refresh functions.

The minimum database required to run the benchmark holds business data from 10,000 suppliers. It contains almost 10 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 metrics 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 (see Clause 5.4.6). 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.

Benchmarks 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 specific customer application benchmarking when critical capacity planning and/or product evaluation decisions are contemplated.

1 General Items

1.1 Benchmark Sponsor

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

IBM Corporation sponsored this TPC-H benchmark.

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:

- *Database 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.*

Appendix A, "Tunable Parameters," contains a list of all database parameters and operating system parameters.

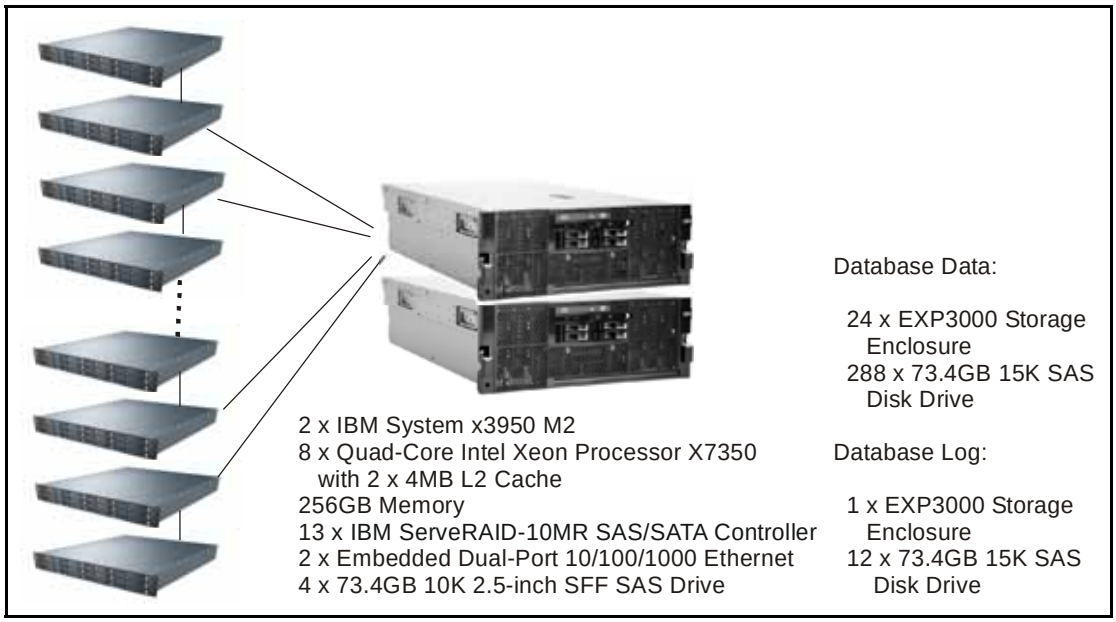
1.3 Configuration Diagrams

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

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

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

1.3.1 Priced and Measured Configurations



The priced configuration for the x3950 M2 contained:

- Eight Intel Xeon Processor X7350 (2.93GHz, 2 x 4MB L2 cache)
- Sixty-four 4GB PC2-5300 DDR2 ECC SDRAM DIMMs
- Two embedded dual-port 10/100/1000 Gigabit Ethernet interfaces (only one was used for the benchmark)
- Thirteen ServeRAID-10M SAS/SATA controllers
- Three hundred (300) 73.4GB 15K 3.5inch SAS disk drives
- Twenty-five (25) EXP3000 Storage Expansion Enclosures
- Four (4) 73.4GB 10K 2.5 inch SAS disk drives

The measured configuration and the priced configuration were identical.

2 Clause 1: Logical Database Design Related Items

2.1 Database Table Definitions

Listings must be provided for all table definition statements and all other statements used to set up the test and qualification databases. (8.1.2.1)

Appendix B contains the scripts that were used to set up the TPC-H test and qualification databases.

2.2 Database Organization

The physical organization of tables and indexes 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.

Appendix B contains the scripts that were used to create the indexes on the test and qualification databases.

2.3 Horizontal Partitioning

Horizontal partitioning of tables and rows in the test and qualification databases must be disclosed (see Clause 1.5.4).

Horizontal partitioning was not used.

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.

3 Clause 2: Queries and Update Functions Related Items

3.1 Query Language

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

SQL was the query language used.

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.

The TPC-supplied DBGEN version 2.6.0 and QGEN version 2.6.0 were used to generate all database populations.

3.3 Substitution Parameters Generation

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 2.6.0 was used to generate the substitution parameters.

3.4 Query Text and Output Data from 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 output for each of the qualification queries. The functional query definitions and variants used in this disclosure use the following minor query modifications:

- In Q1, Q4, Q5, Q6, Q10, Q12, Q14, Q15 and Q20, the "dateadd" function is used to perform date arithmetic.
- In Q7, Q8 and Q9, the "datepart" function is used to extract part of a date (e.g., "YY").
- In Q2, Q3, Q10, Q18 and Q21, the "top" function is used to restrict the number of output rows.
- The word GO is used as a command delimiter.

3.5 Query Substitution Parameters and Seeds Used

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

Appendix C contains the seed and query substitution parameters used.

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 isolation level used to run the queries was "read committed."

3.7 Refresh Function Implementation

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

The refresh functions are part of the implementation-specific layer/driver code included in Appendix D, “Refresh Function Source Code.”

4 Clause 3: Database System Properties Related Items

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.

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

All ACID tests were conducted according to specifications. The Atomicity, Isolation, Consistency and Durability tests were performed on the IBM System x3950 M2 server. Appendix E contains the ACID transaction source code.

4.1.1 Atomicity of Completed Transactions

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

The following steps were performed to verify the Atomicity of completed transactions.

1. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID Transaction was executed using the order key used in step 1.
3. The ACID Transaction committed.
4. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for the same order key. 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, submitting a ROLLBACK of the transaction
for the COMMIT of the transaction. Verify that the appropriate rows have not been changed in the ORDER, LINEITEM, and HISTORY tables.

The following steps were performed to verify the Atomicity of the aborted ACID transaction:

1. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for a randomly selected order key.
2. The ACID Transaction was performed using the order key from step 1. The transaction was stopped prior to the commit.
3. The ACID Transaction was ROLLED BACK.
4. The total price from the ORDER table and the extended price from the LINEITEM table were retrieved for the same order key used in steps 1 and 2. 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.

A consistent state for the TPC-H database is defined to exist when:

O_TOTALPRICE=SUM(L_EXTENDEDPRICE*(1-L_DISCOUNT)*(1+L_TAX)

4.2.1 Consistency Tests

The following steps were performed to verify consistency:

1. The consistency of the ORDER and LINEITEM tables was verified based on a sample of O_ORDERKEYs.
2. One hundred ACID Transactions were submitted from each of six execution streams.
3. The consistency of the ORDER and LINEITEM tables was reverified.

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

This test demonstrates isolation for the read-write conflict of a read-write transaction and a read-only transaction when the read-write transaction is committed.

The following steps were performed to satisfy the test of isolation for a read-only and a read-write committed transaction:

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 blocked and did not see any uncommitted changes made by the ACID Transaction.
3. The ACID Transaction was resumed and committed.
4. The ACID query completed. It returned the data as committed by the ACID Transaction.

4.3.2 Isolation Test 2 - Read-Write Conflict with Rollback

This test demonstrates isolation for the read-write conflict of read-write transaction and read-only transaction when the read-write transaction is rolled back.

The following steps were performed to satisfy the test of isolation for read-only and a rolled back read-write transaction:

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 any 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

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

The following steps were performed to verify isolation of two update transactions:

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 the same O_KEY and L_KEY and a randomly selected DELTA.
3. T2 waited.

4. The ACID transaction T1 was allowed to Commit and T2 completed.
5. It was verified that:

   T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE
   +(DELTA1*(T1.L_EXTENDEDPRICE/T1.L_QUANTITY))

4.3.4 Isolation Test 4 - Write-Write Conflict with Rollback

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

The following steps were performed to verify the isolation of two update transactions after the first one is rolled back:

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 Rollback.
2. Another ACID Transaction T2 was started using the same O_KEY and L_KEY used in step 1 and a randomly selected DELTA.
3. T2 waited.
4. T1 was allowed to ROLLBACK and T2 completed.
5. It was verified that T2.L_EXTENDEDPRICE = T1.L_EXTENDEDPRICE.

4.3.5 Isolation Test 5 - Concurrent Read and Write Transactions on Different Tables

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

The following steps were performed:

1. An ACID Transaction T1 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.
4. T1 completed and the appropriate rows in the ORDER, LINEITEM and HISTORY tables were changed.

4.3.6 Isolation Test 6 - Update Transactions during Continuous Read-Only Query Stream

This test demonstrates 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.

The following steps were performed:

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 amount of time.
2. Before T1 completed, an ACID Transaction T2 was started using randomly selected values of O_KEY, L_KEY and DELTA.
3. T2 completed before T1 completed.
4. It was verified that the appropriate rows in the ORDER, LINEITEM and HISTORY tables were changed.

4.4 Durability Requirements

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

4.4.1 Loss of Log and Failure of a Durable Medium

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

The database log was stored on RAID-10 protected storage. The tables for the database were stored on RAID-0 storage. Two backups of the database were taken to separate disks on RAID-0 storage.

The tests were conducted on the qualification database. The steps performed are shown below.

1. The complete database was backed up twice to RAID-0 protected storage.
2. Seven streams of ACID transactions were started. Each stream executed a minimum of 100 transactions.
3. While the test was running, one of the disks from the database RAID-10 log was removed.
4. The test continued running for approximately an additional 30 transactions per stream.
5. One physical drive of a RAID-0 data volume was removed.
6. A checkpoint of the Qualification database was issued to cause the database to acknowledge the failure.
7. The Stepmaster program was stopped
8. The failed data disk was replaced with a new drive. The Array containing the failed drive was deleted and recreated. The database data partitions were recreated.
9. The Failed Log disk was replaced. The rebuild of the RAID-10 device started automatically and completed successfully.
10. A database restore was issued using the backup taken at the beginning of this test.
11. A command was issued causing the database to run through its roll-forward recovery.
12. The counts in the success files and the HISTORY table count were compared and were found to match.

4.4.2 System Crash

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.

1. Seven streams of ACID transactions were started from a remote client who had access to the database. Each stream executed a minimum of 100 transactions.
2. Then the system was powered off.
3. When the power was restored, the system rebooted and the database was restarted.
4. The database went through a recovery period.
5. The success file and the HISTORY table counts were compared and found to match

4.4.3 Memory Failure

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

See the previous section.

5 Clause 4: Scaling and Database Population Related Items

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	Rows
Order	450,000,000
Lineitem	1,799,989,091
Customer	45,000,000
Part	60,000,000
Supplier	3,000,000
Partsupp	240,000,000
Nation	25
Region	5

5.2 Distribution of Tables and Logs

The distribution of tables and logs across all media must be explicitly described.

The following series of tables shows the distribution of tables and logs across all media.

Controller	Drives	Logical Node/Partition	Size	Use
LSI 1078 (embedded RAID controller)	2 - 73.4GB RAID-1	C:\	36GB	OS; DB; Kit
		D:\	31GB	Unused
	1-73.4GB	L:\	68.36GB	Pagefile
	1-73.4GB	M:\	68.36GB	Kit Backup
ServeRAID-10MR 1	12-73.4GB RAID-0	Physical disk 3	804.65GB	Extended Partition
		C:\mp\tpch300g1	22.7GB	Database Data
		C:\mp\tempdev1	12.5GB	Temp Tables
		C:\mp\loadfg1	22.7GB	Load FileGroup
		unused	18.0GB	Unused
		E:\	160.0GB	Backup1 and Flatfiles
	12-73.4GB RAID-0	Physical disk 4	804.65GB	Extended Partition
		C:\mp\tpch300g2	22.7GB	Database Data
		C:\mp\tempdev2	12.5GB	Temp Tables
		C:\mp\loadfg2	22.7GB	Load FileGroup
		Unused	18.0GB	Unused
		F:\	160.0GB	Backup1 and Flatfiles
ServeRAID-10MR 2	12-73.4GB RAID-0	Physical disk 5	804.65GB	Extended Partition
		C:\mp\tpch300g3	22.7GB	Database Data
		C:\mp\tempdev3	12.5GB	Temp Tables
		C:\mp\loadfg3	22.7GB	Load FileGroup
		Unused	18.0GB	Unused
		G:\	160.0GB	Backup1 and Flatfiles
	12-73.4GB RAID-0	Physical disk 6	804.65GB	Extended Partition
		C:\mp\tpch300g4	22.7GB	Database Data
		C:\mp\tempdev4	12.5GB	Temp Tables
		C:\mp\loadfg4	22.7GB	Load FileGroup
		Unused	18.0GB	Unused
		H:\	160.0GB	Backup1 and Flatfiles
ServeRAID-10MR 3	12-73.4GB RAID-0	Physical disk 7	804.65GB	Extended Partition
		C:\mp\tpch300g5	22.7GB	Database Data
		C:\mp\tempdev5	12.5GB	Temp Tables
		C:\mp\loadfg5	22.7GB	Load FileGroup
		Unused	18.0GB	Unused
		I:\	160.0GB	Backup1 and Flatfiles
	12-73.4GB RAID-0	Physical disk 8	804.65GB	Extended Partition
		C:\mp\tpch300g6	22.7GB	Database Data
		C:\mp\tempdev6	12.5GB	Temp Tables
		C:\mp\loadfg6	22.7GB	Load FileGroup
		Unused	18.0GB	Unused
		J:\	160.0GB	Backup1 and Flatfiles
ServeRAID-10MR 4	12-73.4GB RAID-0	Physical disk 9	804.65GB	Extended Partition
		C:\mp\tpch300g7	22.7GB	Database Data
		C:\mp\tempdev7	12.5GB	Temp Tables
		C:\mp\loadfg7	22.7GB	Load FileGroup
		Unused	18.0GB	Unused
		O:\	160.0GB	Backup1 and Flatfiles
	12-73.4GB RAID-0	Physical disk 10	804.65GB	Extended Partition
		C:\mp\tpch300g8	22.7GB	Database Data
		C:\mp\tempdev8	12.5GB	Temp Tables
		C:\mp\loadfg8	22.7GB	Load FileGroup
		Unused	18.0GB	Unused
		P:\	160.0GB	Backup1 and Flatfiles

Controller	Drives	Logical Node/Partition	Size	Use
ServeRAID-10MR 5	12-73.4GB RAID-0	Physical disk 11 C:\mp\tpch300g9 C:\mp\tempdev9 C:\mp\loadfg9 Unused Q:\ C:\mp\tpch1g1 C:\mp\loadfg1g_1	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB 160.0GB 20GB 20GB	Extended Partition Database Data Temp Tables Load FileGroup Unused Backup2 Qualification Database Data Qualification LoadFG
	12-73.4GB RAID-0	Physical disk 12 C:\mp\tpch300g10 C:\mp\tempdev10 C:\mp\loadfg10 Unused R:\ C:\mp\tpch1g2 C:\mp\loadfg1g_2	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB 160.0GB 20GB 20GB	Extended Partition Database Data Temp Tables Load FileGroup Unused Backup2 Qualification Database Data Qualification LoadFG
ServeRAID-10MR 6	12-73.4GB RAID-0	Physical disk 13 C:\mp\tpch300g11 C:\mp\tempdev11 C:\mp\loadfg11 Unused S:\	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB 160.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused Backup2
	12-73.4GB RAID-0	Physical disk 14 C:\mp\tpch300g12 C:\mp\tempdev12 C:\mp\loadfg12 Unused T:\	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB 160.0	Extended Partition Database Data Temp Tables Load FileGroup Unused Backup2
ServeRAID-10MR 7	12-73.4GB RAID-10	Physical disk 15 C:\mp\tpch300glog C:\mp\templog1 N:\RF_Flatfiles (NTFS) C:\mp\tpch1glog	402.32GB 100GB 14.65GB 100GB 500MB	Extended Partition Database Log Temp Database Log RF Flatfiles 1GB Database Log
ServeRAID-10MR 8	12-73.4GB RAID-0	Physical disk 16 C:\mp\tpch300g13 C:\mp\tempdev13 C:\mp\loadfg13 Unused U:\	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB 160.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused Backup2
	12-73.4GB RAID-0	Physical disk 17 C:\mp\tpch300g14 C:\mp\tempdev14 C:\mp\loadfg14 Unused V:\	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB 160.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused Backup2
ServeRAID-10MR 9	12-73.4GB RAID-0	Physical disk 18 C:\mp\tpch300g15 C:\mp\tempdev15 C:\mp\loadfg15 Unused W:\	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB 160.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused Backup2
	12-73.4GB RAID-0	Physical disk 19 C:\mp\tpch300g16 C:\mp\tempdev16 C:\mp\loadfg16 Unused X:\	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB 160.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused Backup2

Controller	Drives	Logical Node/Partition	Size	Use
ServeRAID-10MR 10	12-73.4GB RAID-0	Physical disk 20 C:\mp\tpch300g17 C:\mp\tempdev17 C:\mp\loadfg17 Unused	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB	Extended Partition Database Data Temp Tables Load FileGroup unused
	12-73.4GB RAID-0	Physical disk 21 C:\mp\tpch300g18 C:\mp\tempdev18 C:\mp\loadfg18 Unused	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused
ServeRAID-10MR 11	12-73.4GB RAID-0	Physical disk 22 C:\mp\tpch300g19 C:\mp\tempdev19 C:\mp\loadfg19 Unused	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused
	12-73.4GB RAID-0	Physical disk 23 C:\mp\tpch300g20 C:\mp\tempdev20 C:\mp\loadfg20 Unused	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused
ServeRAID-10MR 12	12-73.4GB RAID-0	Physical disk 24 C:\mp\tpch300g21 C:\mp\tempdev21 C:\mp\loadfg21 Unused	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused
	12-73.4GB RAID-0	Physical disk 25 C:\mp\tpch300g22 C:\mp\tempdev22 C:\mp\loadfg22 Unused	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused
ServeRAID-10MR 13	12-73.4GB RAID-0	Physical disk 26 C:\mp\tpch300g23 C:\mp\tempdev23 C:\mp\loadfg23 Unused	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused
	12-73.4GB RAID-0	Physical disk 27 C:\mp\tpch300g24 C:\mp\tempdev24 C:\mp\loadfg24 Unused	804.65GB 22.7GB 12.5GB 22.7GB 18.0GB	Extended Partition Database Data Temp Tables Load FileGroup Unused

5.3 Database Partition / Replication Mapping

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

The database was not replicated or partitioned.

5.4 RAID Implementation

Implementations may use some form of RAID to ensure high availability. If used for data, auxiliary storage (e.g., indexes) or temporary space, the level of RAID must be disclosed for each device.

RAID-10 was used for log disks. RAID-1 was used for the Operating System/Database install disk. RAID-0 was used for all other database disks, the temporary tablespace, and the OS swap space.

5.5 DBGEN Modifications

Any modifications to the DBGEN (see Clause 4.2.1) source code must be disclosed. In the event that a program other than DBGEN was used to populate the database, it must be disclosed in its entirety.

The standard distribution DBGEN version 2.6.0 was used for database population. No modifications were made.

5.6 Database Load Time

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

See the Executive Summary at the beginning of this report.

5.7 Data Storage Ratio

The data storage ratio must be disclosed. It is computed as the ratio between the total amount of priced disk space and the chosen test database size as defined in Clause 4.1.3.

The calculation of the data storage ratio is shown in the following table.

Disk Type	Number of Disks	Space per Disk	Total Disk Space	Scale Factor	Storage Ratio
73.4GB 10K 2.5 inch SAS SFF H/S Drive	4	73.4GB	294GB		
73.4GB 10K 3.5 inch SAS H/S Drive	300	73.4GB	22020GB		
Total			22314GB	300GB	74.38

The data storage ratio is 74.38, derived by dividing 22314GB by the database size of 300GB.

5.8 Database Load Mechanism Details and Illustration

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.

Flat files for each of the tables were created using DBGEN.

The tables were loaded as depicted in Figure 4-1.

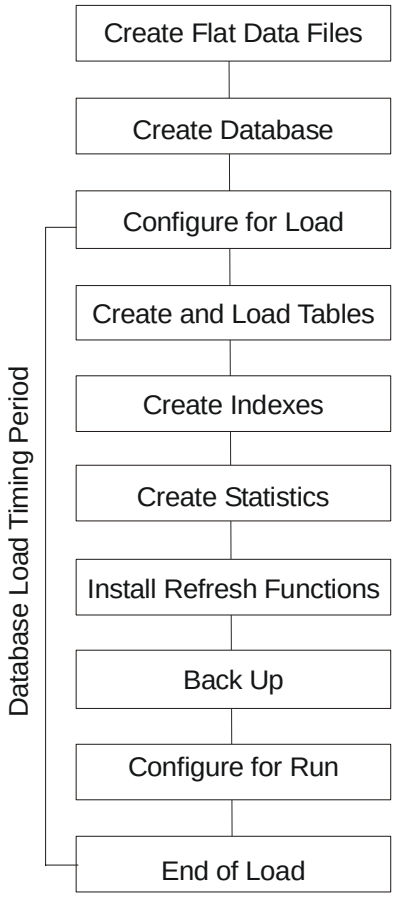


Figure 4-1. Database Load Procedure

5.9 Qualification Database Configuration

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

The qualification database used identical scripts and disk structure to create and load the data with adjustments for size difference. See Section 5.2 for details.

6 Clause 5: Performance Metrics and Execution Rules Related Items

6.1 System Activity between Load and Performance Tests

Any system activity on the SUT that takes place between the conclusion of the load test and the beginning of the performance test must be fully disclosed.

The auditor requested that queries be run against the database to verify the correctness of the database load. The system was rebooted and SQL Server was restarted.

6.2 Steps in the Power Test

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

The following steps were used to implement the power test:

1. RF1 Refresh Transaction
2. Stream 00 Execution
3. RF2 Refresh Transaction

6.3 Timing Intervals for Each Query and Refresh Function

The timing intervals for each query of the measured set and for both update functions must be reported for the power test.

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.4 Number of Streams for the Throughput Test

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

Six streams were used for the throughput test.

6.5 Start and End Date/Times for Each Query Stream

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.6 Total Elapsed Time for the Measurement Interval

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report..

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

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

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

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.9 Performance Metrics

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

See the Numerical Quantities Summary in the Executive Summary at the beginning of this report.

6.10 Performance Metric and Numerical Quantities from Both Runs

The performance metric and numerical quantities from both runs must be disclosed.

Two consecutive runs of the TPC-H benchmark were performed. The following table contains the results for both runs.

	QppH @ 300GB	QthH @ 300GB	QphH @ 300GB
Run1	58,196.9	36,413.8	46,034.4
Run2	59,289.7	36,358.1	46,429.1

6.11 System Activity between Tests

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

The system was rebooted and SQL Server was restarted.

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 functionality or environmental setting on which it relies must be provided. All related source code, scripts and configurations must be disclosed. The information provided should be sufficient for an independent reconstruction of the driver.

Two scripts were used. The first one was used to create and load the database; the second was used to run the Power and Throughput tests. These scripts are in Appendix D. The C program, semaphore.c, was used for coordination of parallel processes.

7.2 Implementation-Specific Layer

If an implementation-specific layer is used, then a detailed description of how it performs its functions must be supplied, including any related source code or scripts. This description should allow an independent reconstruction of the implementation-specific layer.

A command script was used to control and track the execution of queries. The scripts are contained in Appendix D. Qgen was used to generate the query streams, along with the appropriate substitution values.

The following steps are performed, to accomplish the Power and Throughput Runs:

Power Run

1. Execute 32 concurrent RF1 processes, each of which will apply a segment of an update set generated by dbgen.
2. Each process submits multiple transactions, where a transaction spans a set of orders and their associated line items.
3. Execute the Stream0 queries, in the prescribed order.
4. Execute 32 concurrent RF2 processes, each of which will apply a segment of an update set generated by dbgen. Each thread submits multiple transactions, where a transaction spans a set of orders and their associated line items.

Throughput Run

1. Execute six concurrent query streams. Each stream executes queries in the prescribed order for the appropriate Stream Id (1-6). Upon completion of each stream, a semaphore is set to indicate completion.
2. Execute five consecutive RF1/RF2 transactions, against ascending Update sets produced by dbgen. The first RF1 waits on a semaphore prior to beginning its insert operations.

Each step is timed by the script. The timing information is stored in the database for later analysis. The inputs and outputs of steps are stored in text files for later analysis.

7.3 Profile-Directed Optimization

Profile-directed optimization was not used.

8 Clause 7: Pricing Related Items

8.1 Hardware and Software Components

A detailed list of the hardware and software used in the priced system must be reported. Each item must have a vendor part number, description and release/revision level, and either 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) must also be reported.

A detailed list of all hardware and software, including the 3-year price, is provided in the Executive Summary at the front of this report. The price quotations are included in Appendix F.

8.2 Three-Year Cost of System Configuration

The total 3-year price of the entire configuration must be reported, including hardware, software and maintenance charges. Separate component pricing is recommended. The basis of all discounts must be disclosed.

A detailed list of all hardware and software, including the 3-year price, is provided in the Executive Summary at the front of this report. The price quotations are included in Appendix F.

8.3 Availability Dates

The committed delivery date for general availability (availability date) of products used in the price calculations must be reported. When the priced system includes products with different availability dates, availability date reported on 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 7.3.1.3).

All system hardware and software is generally available except for the IBM System x3950 M2 and the IBM ServeRAID-MR10 SAS/SATA controller as detailed in the table below. The Total System Availability Date is January 31, 2008.

Description	Part Number	Order Date	Availability Date	Order Method	Price Verification
IBM System x3950 M2 with Intel Xeon Processor X7350	7141-4SU	1-15-08	1-31-08	See Note 1	See Note 2
IBM ServeRAID-MR10 SAS/SATA Controller	43W4339	10-16-07	12-17-07	See Note 1	N/A

Note 1: IBM – 1-888-746-7426, ext. 5821

Note 2: This component is not immediately orderable. For price verification before order date, call 1-888-746-7426, ext. 5821.

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.

Clause 9: Audit Related Items

9.1 Auditor's Report

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. For a copy of this disclosure, go to www.tpc.org.

INFO SIZING

TPC TRANSACTION PROCESSING PERFORMANCE COUNCIL CERTIFIED AUDITOR

Benchmark Sponsor: Celia Schreiber, Manager
Modular Systems and Blade Performance Analysis and Benchmarking
IBM Systems and Technology Group
3039 Cornwallis Road
Research Triangle Park NC, 27709

February 18, 2008

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

Platform: **IBM® System x™ 3950 M2**

Database Manager: **Microsoft® SQL Server 2005 Enterprise x64 Edition**

Operating System: **Microsoft Windows® Server 2003 R2 Enterprise x64 Edition SP2**

The results were:

CPU (Speed)	Memory	Disks	**QphH@300GB**
IBM® System x™ 3950 M2			
8 x Intel Xeon X7350 (2.93 GHz)	4 MB Cache/2-core 32 GB Main	4 x 73.4 GB 10K 300 x 73.4 GB 15K	**46,034.4**

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 and one query variant
- 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:

None.

Respectfully Yours,

François Raab
President

Appendix A: Tunable Parameters

Microsoft SQL Server 2005 Version

Microsoft SQL Server 2005 - 9.00.3042.00 (X64)
Feb 10 2007 00:59:02
Copyright (c) 1988-2005 Microsoft Corporation
Enterprise Edition (64-bit) on Windows NT 5.2 (Build 3790: Service Pack 2)

Microsoft SQL Server 2005 Startup Parameters

SQLSERVR -x -c -E -T834 -T2301

-x Disable the keeping of CPU time and Cache-hit ratio statistics
-c Start SQL Server independently of Windows NT Service Control Manager
-E Increase the number of consecutive extents allocated per file to four.
-T834 Causes SQL Server to use the Windows large-page allocations for memory allocated to the buffer pool.
-T2301 Enables more accurate query runtime behavior modeling in the SQL Server query optimizer typically only needed for large dataset decision support processing.
-T3502 Show timing of checkpoints in the log

Microsoft SQL Server 2005 Installation

Default options were selected during the installation of SQL Server. Mixed Mode Authenication and Latin1_General binary sort order was specified. Service Packs 1 and 2 were installed.

Microsoft SQL Server 2005 Configuration Parameters

```
name                                minimum     maximum     config_value
run_value
----------------------------------- ----------- ----------- ------------
-----------
Ad Hoc Distributed Queries                    0           1           0
0
affinity I/O mask                  -2147483648  2147483647           0
0
affinity mask                      -2147483648  2147483647          -1
-1
affinity64 I/O mask                -2147483648  2147483647           0
0
affinity64 mask                    -2147483648  2147483647           0
0
Agent XPs                                     0           1           0
0
allow updates                                 0           1           1
1
awe enabled                                   0           1           0
0
blocked process threshold                     0       86400           0
0
c2 audit mode                                 0           1           0
0
clr enabled                                   0           1           0
0
common criteria compliance 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      252000
252000
max text repl size (B)                        0  2147483647       65536
65536
max worker threads                          128       32767        1024
1024
media retention                               0         365           0
0
min memory per query (KB)                   512  2147483647         512
512
min server memory (MB)                        0  2147483647           0
16
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           0
0
query governor cost limit                     0  2147483647           0
0
query wait (s)                               -1  2147483647  2147483647
2147483647
recovery interval (min)                       0       32767       32767
32767
remote access                                 0           1           1
1
remote admin connections                      0           1           0
0
remote login timeout (s)                      0  2147483647           0
0
remote proc trans                             0           1           0
0
remote query timeout (s)                      0  2147483647           0
```

0
Replication XPs 0 1 0
0
scan for startup procs 0 1 0
0
server trigger recursion 0 1 1
1
set working set size 0 1 0
0
show advanced options 0 1 1
1
SMO and DMO XPs 0 1 1
1
SQL Mail XPs 0 1 0
0
transform noise words 0 1 0
0
two digit year cutoff 1753 9999 2049
2049
user connections 0 32767 0
0
user options 0 32767 0
0
Web Assistant Procedures 0 1 0
0
xp_cmdshell 0 1 0
0

Microsoft Windows Server 2003 Configuration

Windows 2003 COnfiguration

The default installation of Windows 2003 Enterprise Endition was used. A TCP/IP addressed used to access
the machine remotely was configured during the install.

Installation was updated with the following change:
Control Panel> System Properties> Advanced> Performance Options> Programs.

Windows Registry Modifications

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 8:58 AM

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration\Node0
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 8:58 AM
Value 0
Name: CPUMask
Type: REG_DWORD
Data: 0xf

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration\Node1
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 8:59 AM
Value 0
Name: CPUMask
Type: REG_DWORD
Data: 0xf0

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration\Node2
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 8:59 AM
Value 0
Name: CPUMask
Type: REG_DWORD
Data: 0xf00

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration\Node3
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 8:59 AM
Value 0
Name: CPUMask
Type: REG_DWORD
Data: 0xf000

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration\Node4
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 9:00 AM
Value 0
Name: CPUMask
Type: REG_DWORD
Data: 0xf0000

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration\Node5
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 9:00 AM
Value 0
Name: CPUMask
Type: REG_DWORD
Data: 0xf00000

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration\Node6
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 9:01 AM
Value 0
Name: CPUMask
Type: REG_DWORD
Data: 0xf000000

Key Name: HKEY_LOCAL_MACHINE\SOFTWARE\Microsoft\Microsoft SQL Server\90\NodeConfiguration\Node7
Class Name: <NO CLASS>
Last Write Time: 10/26/2007 - 9:01 AM
Value 0
Name: CPUMask
Type: REG_DWORD
Data: 0xf0000000

Windows Registry Editor Version 5.00

[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management]
"ClearPageFileAtShutdown"=dword:00000000
"DisablePagingExecutive"=dword:00000001
"LargeSystemCache"=dword:00000000
"NonPagedPoolQuota"=dword:00000000
"NonPagedPoolSize"=dword:00000000
"PagedPoolQuota"=dword:00000000
"PagedPoolSize"=dword:00000000
"SecondLevelDataCache"=dword:00000000
"SystemPages"=dword:00000000
"PagingFiles"=hex(7):6c,00,3a,00,5c,00,70,00,61,00,67,00,65,00,66,00,69,00,6c,\
00,65,00,2e,00,73,00,79,00,73,00,20,00,31,00,36,00,33,00,38,00,34,00,20,00,\
36,00,35,00,35,00,33,00,36,00,00,00,00,00
"PhysicalAddressExtension"=dword:00000001
"WriteWatch"=dword:00000001
"DontVerifyRandomDrivers"=dword:00000001

Key Name:
HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\I/O System
Class Name: <NO CLASS>
Last Write Time: 8/13/2007 - 6:34 PM
Value 0
Name: CountOperations
Type: REG_DWORD
Data: 0

ServeRAID-MR10 Configuration

```
==============================================================
=================
Adapter: 0
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L002353407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930335
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ101V81130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931379
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ1011Y1130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049327dd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ1006K1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000493133d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ101091130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931875
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ100WY1130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
```

Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930e6d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ100N61130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931849
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ100VB1130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004932b49
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Y24Q1130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930349
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101VF1130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931e01
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101G81130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004932ab1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QKBD1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004932b59
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QETY1130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0

Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000491dff1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Y4G71130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000491d2f5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Y5HH1130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000492649d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Z9AW1130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049164a5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QFT41130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004926025
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QFAM1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000491e2e5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Y4M11130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004926171
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Z6PB1130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925d79
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0V1FT1130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]

```
Firmware state: Online
SAS Address(0): 0x5000c50001c3f459
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SFZ61130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049250d9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0V2SR1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004926015
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Y5N31130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925fdd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0V1H41130BA23

==============================================================
=================
Adapter: 1
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L001373407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 85
Slot Number: 1
Device Id: 73
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4befd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCH61130BA23

Physical Disk: 1
Enclosure Device ID: 85
Slot Number: 2
Device Id: 74
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bf89
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCJ71130BA23

Physical Disk: 2
Enclosure Device ID: 85
Slot Number: 3
Device Id: 75
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba61
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCDN1130BA23
```

Physical Disk: 3
Enclosure Device ID: 85
Slot Number: 4
Device Id: 76
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bf71
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCHR1130BA23

Physical Disk: 4
Enclosure Device ID: 85
Slot Number: 5
Device Id: 77
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b931
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCC41130BA23

Physical Disk: 5
Enclosure Device ID: 85
Slot Number: 6
Device Id: 78
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bba5
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCER1130BA23

Physical Disk: 6
Enclosure Device ID: 85
Slot Number: 7
Device Id: 79
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c462d1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZYB1130BA23

Physical Disk: 7
Enclosure Device ID: 85
Slot Number: 8
Device Id: 80
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4be01
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0QJ1130BA23

Physical Disk: 8
Enclosure Device ID: 85
Slot Number: 9
Device Id: 81
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bc19
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R29B1130BA23

Physical Disk: 9
Enclosure Device ID: 85
Slot Number: 10
Device Id: 82
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b725
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R2WF1130BA23

Physical Disk: 10
Enclosure Device ID: 85
Slot Number: 11
Device Id: 83
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b639
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SC981130BA23

Physical Disk: 11
Enclosure Device ID: 85
Slot Number: 12
Device Id: 84

```
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b915
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCC01130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 72
Slot Number: 1
Device Id: 60
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b685
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC9T1130BA23

Physical Disk: 1
Enclosure Device ID: 72
Slot Number: 2
Device Id: 61
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bd35
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0LSCY1130BA23

Physical Disk: 2
Enclosure Device ID: 72
Slot Number: 3
Device Id: 62
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b8dd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R3XP1130BA23

Physical Disk: 3
Enclosure Device ID: 72
Slot Number: 4
Device Id: 63
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bbf5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCBE1130BA23

Physical Disk: 4
Enclosure Device ID: 72
Slot Number: 5
Device Id: 64
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba7d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCE11130BA23

Physical Disk: 5
Enclosure Device ID: 72
Slot Number: 6
Device Id: 65
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b705
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCAB1130BA23

Physical Disk: 6
Enclosure Device ID: 72
Slot Number: 7
```

```
Device Id: 66
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bbb5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCEW1130BA23

Physical Disk: 7
Enclosure Device ID: 72
Slot Number: 8
Device Id: 67
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bd49
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1PM1130BA23

Physical Disk: 8
Enclosure Device ID: 72
Slot Number: 9
Device Id: 68
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b8d1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC951130BA23

Physical Disk: 9
Enclosure Device ID: 72
Slot Number: 10
Device Id: 69
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b7ed
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R07P1130BA23

Physical Disk: 10
Enclosure Device ID: 72
Slot Number: 11
Device Id: 70
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bdfd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCFY1130BA23

Physical Disk: 11
Enclosure Device ID: 72
Slot Number: 12
Device Id: 71
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47955
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0XE1130BA23

==============================================================
=================
Adapter: 2
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L000313407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
```

Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930a55
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QF4F1130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049303cd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101Y31130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930525
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101PZ1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930561
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101QN1130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931659
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ1013M1130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049301f9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101S51130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930415
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101W81130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049312ed
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ100W01130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online

SAS Address(0): 0x5000c500049310b9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ100RT1130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930421
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101YY1130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931111
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ100SD1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049302cd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ101TP1130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000492ce71
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Y7HB1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000491e1b1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Y4L21130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000491e395
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Y4PK1130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]

```
Firmware state: Online
SAS Address(0): 0x5000c50004925e95
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0YWXQ1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004932371
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0ZZYW1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000493255d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ1002E1130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925e9d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0V2T71130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004926001
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0V1HH1130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931a79
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ101AA1130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931be9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ101CB1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049313d5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ100NN1130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925a59
SAS Address(1): 0x0
Connected Port Number: 4(path0)
```

```
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0V38N1130BA23

==============================================================
=================
Adapter: 3
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L001073407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b669
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC9G1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b911
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC8C1130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c479b1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0WR1130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b989
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCCB1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47d29
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1JK1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b9e1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCCX1130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
```

Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47cbd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R1G21130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bea1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCGT1130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c465f1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7V81130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bae1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R2HM1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4baf9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCEK1130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bd7d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZGW1130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b741
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZPC1130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2

Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b991
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCCK1130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b721
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R2KG1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47621
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7M11130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b675
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SC9R1130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b885
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZG61130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b535
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SC861130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bc4d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SGCR1130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b8b9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCB91130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0

```
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b9b5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R3ST1130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4becd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCGW1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47c3d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1HX1130BA23

==============================================================
=================
Adapter: 4
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L002383407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004926071
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0V1HN1130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000492504d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0V3K81130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049255e9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Z5DT1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
```

SAS Address(0): 0x5000c50004925fed
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0V1H61130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049250f1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0V2JG1130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925d25
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QLND1130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925d81
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Z9SN1130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925e2d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0ST321130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925169
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SVX11130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049256dd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Z5BE1130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925ef9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0V1GT1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925c31
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0V33L1130BA23

```
DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000491e6c9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Y32D1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000491e5ad
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Y31F1130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925199
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0V2RM1130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004931b6d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ101CG1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000491e671
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Y32R1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925ea1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0QHH01130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049327bd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
```

```
Inquiry Data: IBM-ESXSST373455SS     BA233LQ1005Z1130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930649
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0VG5T1130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925029
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0V3K51130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004925f4d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0XXTR1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c5000493253d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ100281130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c500049317dd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ1016P1130BA23

==============================================================
=================
Adapter: 5
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L001243407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba21
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCDE1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
```

Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b881
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R6WV1130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bee1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCH21130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b8e5
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCBH1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bfc1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCJS1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c453d9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Q0VK1130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bdb1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0JLY81130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bef5
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCH41130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b7d5
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCB31130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0

```
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b7b1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1LQ1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b775
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0QAZ31130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46751
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R7Y21130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba01
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCD31130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba09
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCD61130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b6b1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0PD1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bbad
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC6Z1130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
```

```
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4baed
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCEG1130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b635
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SC971130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b6f9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCAG1130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bfad
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCJH1130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b6e9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCAC1130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b9d5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCCW1130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b7ad
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCAW1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba85
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCDT1130BA23

==============================================================
=================
Adapter: 6
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L001283407
==============================================================
=================
Number of DISK GROUPS: 1
```

DISK GROUPS: 0
Number of Spans: 6
SPAN: 0
Span Reference: 0x00
Number of PDs: 2
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-1, Secondary-3, RAID Level Qualifier-0
Size:411984MB
State: Optimal
Stripe Size: 64kB
Number Of Drives:2
Span Depth:6
Default Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Current Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46475
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R85P1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47989
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7RV1130BA23

SPAN: 1
Span Reference: 0x01
Number of PDs: 2
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-1, Secondary-3, RAID Level Qualifier-0
Size:411984MB
State: Optimal
Stripe Size: 64kB
Number Of Drives:2
Span Depth:6
Default Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Current Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f291
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Q0ZR1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46c75
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7ZX1130BA23

SPAN: 2
Span Reference: 0x02
Number of PDs: 2
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-1, Secondary-3, RAID Level Qualifier-0
Size:411984MB
State: Optimal
Stripe Size: 64kB
Number Of Drives:2
Span Depth:6
Default Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Current Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]

Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45bd1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R24F1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46675
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7WQ1130BA23

SPAN: 3
Span Reference: 0x03
Number of PDs: 2
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-1, Secondary-3, RAID Level Qualifier-0
Size:411984MB
State: Optimal
Stripe Size: 64kB
Number Of Drives:2
Span Depth:6
Default Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Current Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bfc5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCJL1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c460bd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R1561130BA23

SPAN: 4
Span Reference: 0x04
Number of PDs: 2
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-1, Secondary-3, RAID Level Qualifier-0
Size:411984MB
State: Optimal
Stripe Size: 64kB
Number Of Drives:2
Span Depth:6
Default Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Current Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47ced
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R6TK1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4694d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7YG1130BA23

SPAN: 5
Span Reference: 0x05
Number of PDs: 2
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:

```
RAID Level: Primary-1, Secondary-3, RAID Level Qualifier-0
Size:411984MB
State: Optimal
Stripe Size: 64kB
Number Of Drives:2
Span Depth:6
Default Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad
BBU
Current Cache Policy: WriteThrough, ReadAdaptive, Direct, No Write Cache if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bf7d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCJ51130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45c41
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0QZQT1130BA23

==============================================================
=================
Adapter: 7
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L000503407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c40df9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SG4H1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bf05
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCH71130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bcd5
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R3SQ1130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
```

Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47a19
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7SN1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45d79
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZSW1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bf3d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCHK1130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b71d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCAK1130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b8f1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SC841130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45cf9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZZ61130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bb01
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCEH1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b759
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7YY1130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba6d
SAS Address(1): 0x0

```
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R83D1130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b801
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R2YW1130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b8e9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC7W1130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b901
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC7F1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46671
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R7WN1130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bf91
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCHH1130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c471fd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0M31130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46131
```

```
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0QZNA1130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b63d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0R2L41130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b845
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0QZLG1130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b561
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SC871130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bc55
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCSA1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b8cd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SC6V1130BA23

==============================================================
=================
Adapter: 8
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L002493407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bd6d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS     BA233LQ0SCB61130BA23

Physical Disk: 1
Enclosure Device ID: 20
```

Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46361
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R1C71130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47ba1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7TS1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c463c1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R2571130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c478e5
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0S21130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47839
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7QH1130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46075
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0NX1130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46975
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZLR1130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba0d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCD41130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0

```
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f901
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0P0XL1130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c461ad
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R14J1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45d39
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R08X1130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f429
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SG3V1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46a75
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1QF1130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b649
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC991130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b7c1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SC7Q1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
```

```
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46e79
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1121130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c424dd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Q0E91130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bedd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCH11130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b70d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCAD1130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46cbd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R8051130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b9c1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCCQ1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45c5d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0QZRE1130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47b7d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R6RW1130BA23

==============================================================
=================
Adapter: 9
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L002343407
==============================================================
=================
```

```
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4625d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1741130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47ae9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1JT1130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b82d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0FH1130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c474fd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R7JX1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4735d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R7GE1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bd21
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SENG1130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b69d
SAS Address(1): 0x0
```

Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SC9S1130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46111
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0671130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4772d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7NE1130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c467f9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R13M1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46d81
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R81M1130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bee5
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCH31130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 46
Slot Number: 1
Device Id: 34
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba89
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCE51130BA23

Physical Disk: 1
Enclosure Device ID: 46
Slot Number: 2
Device Id: 35
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46561

SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R86Q1130BA23

Physical Disk: 2
Enclosure Device ID: 46
Slot Number: 3
Device Id: 36
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b975
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCCG1130BA23

Physical Disk: 3
Enclosure Device ID: 46
Slot Number: 4
Device Id: 37
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bea9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0VX1130BA23

Physical Disk: 4
Enclosure Device ID: 46
Slot Number: 5
Device Id: 38
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4beb1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0631130BA23

Physical Disk: 5
Enclosure Device ID: 46
Slot Number: 6
Device Id: 39
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b94d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCCE1130BA23

Physical Disk: 6
Enclosure Device ID: 46
Slot Number: 7
Device Id: 40
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4be75
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCGR1130BA23

Physical Disk: 7
Enclosure Device ID: 46
Slot Number: 8
Device Id: 41
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47469
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7H11130BA23

Physical Disk: 8
Enclosure Device ID: 46
Slot Number: 9
Device Id: 42
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4be51
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCGA1130BA23

Physical Disk: 9
Enclosure Device ID: 46
Slot Number: 10
Device Id: 43
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b655
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0QS1130BA23

```
Physical Disk: 10
Enclosure Device ID: 46
Slot Number: 11
Device Id: 44
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4be61
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCGK1130BA23

Physical Disk: 11
Enclosure Device ID: 46
Slot Number: 12
Device Id: 45
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4be25
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCG61130BA23

==============================================================
=================
Adapter: 10
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L000723407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f4dd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Q0GB1130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f2c1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Q0ZG1130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4623d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R17H1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4baa1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCE21130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
```

Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45d59
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZTR1130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45c4d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZS81130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bf35
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCHF1130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46335
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZE61130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3fadd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Q3JW1130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47f7d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R6WG1130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c469b9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R1ZG1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46929
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7X31130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:

RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46889
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZLC1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b719
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCAH1130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46dbd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R8241130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46969
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0ZS1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f64d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Q0KJ1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b7f1
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R2Y91130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47f71
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R1P61130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0

```
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46f11
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R8321130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c477b9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R7PJ1130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47221
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0LW1130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46509
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1381130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46575
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R10B1130BA23

==============================================================
=================
Adapter: 11
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L001263407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 46
Slot Number: 1
Device Id: 34
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba71
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCDQ1130BA23

Physical Disk: 1
Enclosure Device ID: 46
Slot Number: 2
Device Id: 35
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bfa1
SAS Address(1): 0x0
```

Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCJF1130BA23

Physical Disk: 2
Enclosure Device ID: 46
Slot Number: 3
Device Id: 36
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bad9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCEB1130BA23

Physical Disk: 3
Enclosure Device ID: 46
Slot Number: 4
Device Id: 37
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b7c9
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R2JG1130BA23

Physical Disk: 4
Enclosure Device ID: 46
Slot Number: 5
Device Id: 38
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b9f1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCCY1130BA23

Physical Disk: 5
Enclosure Device ID: 46
Slot Number: 6
Device Id: 39
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bc89
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0BTXS1130BA23

Physical Disk: 6
Enclosure Device ID: 46
Slot Number: 7
Device Id: 40
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46489
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R13K1130BA23

Physical Disk: 7
Enclosure Device ID: 46
Slot Number: 8
Device Id: 41
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bf8d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SCJB1130BA23

Physical Disk: 8
Enclosure Device ID: 46
Slot Number: 9
Device Id: 42
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4baf1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R3R11130BA23

Physical Disk: 9
Enclosure Device ID: 46
Slot Number: 10
Device Id: 43
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c469ad
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0ZG1130BA23

Physical Disk: 10

```
Enclosure Device ID: 46
Slot Number: 11
Device Id: 44
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45c7d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0QZWG1130BA23

Physical Disk: 11
Enclosure Device ID: 46
Slot Number: 12
Device Id: 45
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b661
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0HNAL1130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, No Write Cache if Bad BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, No Write Cache if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4b829
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1GX1130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47f35
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1PS1130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bb7d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1DQ1130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bae5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCEJ1130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46441
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0KY1130BA23

Physical Disk: 5
Enclosure Device ID: 20
```

```
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c469fd
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1Q61130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47d41
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R1GV1130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47615
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R7LX1130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47891
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0RE1130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4bfa5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCJ91130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4ba91
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SCDV1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 7
Device Id: 47
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50004930589
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ101NE1130BA23

==============================================================
=================
Adapter: 12
Product Name: SystemX MegaRAID SAS 8880EM2
Memory: 256MB
BBU: Present
Serial No: L003373407
==============================================================
=================
Number of DISK GROUPS: 2

DISK GROUPS: 0
Number of Spans: 1
SPAN: 0
Span Reference: 0x00
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 0)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
```

```
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad
BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 20
Slot Number: 1
Device Id: 8
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f655
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Q0XJ1130BA23

Physical Disk: 1
Enclosure Device ID: 20
Slot Number: 2
Device Id: 9
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47545
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R7KP1130BA23

Physical Disk: 2
Enclosure Device ID: 20
Slot Number: 3
Device Id: 10
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c40dcd
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SG781130BA23

Physical Disk: 3
Enclosure Device ID: 20
Slot Number: 4
Device Id: 11
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3fab1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0Q0KR1130BA23

Physical Disk: 4
Enclosure Device ID: 20
Slot Number: 5
Device Id: 12
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4612d
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R0SH1130BA23

Physical Disk: 5
Enclosure Device ID: 20
Slot Number: 6
Device Id: 13
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f741
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0SFQN1130BA23

Physical Disk: 6
Enclosure Device ID: 20
Slot Number: 7
Device Id: 14
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46351
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS      BA233LQ0R23T1130BA23

Physical Disk: 7
Enclosure Device ID: 20
Slot Number: 8
Device Id: 15
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
```

Firmware state: Online
SAS Address(0): 0x5000c50001c403b1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SFWN1130BA23

Physical Disk: 8
Enclosure Device ID: 20
Slot Number: 9
Device Id: 16
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c45ee1
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0S51130BA23

Physical Disk: 9
Enclosure Device ID: 20
Slot Number: 10
Device Id: 17
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46f45
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R83B1130BA23

Physical Disk: 10
Enclosure Device ID: 20
Slot Number: 11
Device Id: 18
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46875
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZLH1130BA23

Physical Disk: 11
Enclosure Device ID: 20
Slot Number: 12
Device Id: 19
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c475ed
SAS Address(1): 0x0
Connected Port Number: 0(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7LC1130BA23

DISK GROUPS: 1
Number of Spans: 1
SPAN: 0
Span Reference: 0x01
Number of PDs: 12
Number of VDs: 1
Number of dedicated Hotspares: 0
Virtual Disk Information:
Virtual Disk: 0 (target id: 1)
Name:
RAID Level: Primary-0, Secondary-0, RAID Level Qualifier-0
Size:823968MB
State: Optimal
Stripe Size: 128kB
Number Of Drives:12
Span Depth:1
Default Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Current Cache Policy: WriteBack, ReadAdaptive, Direct, Write Cache OK if Bad BBU
Access Policy: Read/Write
Disk Cache Policy: Disk's Default
Physical Disk Information:
Physical Disk: 0
Enclosure Device ID: 33
Slot Number: 1
Device Id: 21
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f38d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0Q0WC1130BA23

Physical Disk: 1
Enclosure Device ID: 33
Slot Number: 2
Device Id: 22
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c460c9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0SD1130BA23

Physical Disk: 2
Enclosure Device ID: 33
Slot Number: 3
Device Id: 23
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]

Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46395
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R0CF1130BA23

Physical Disk: 3
Enclosure Device ID: 33
Slot Number: 4
Device Id: 24
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c47605
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7LT1130BA23

Physical Disk: 4
Enclosure Device ID: 33
Slot Number: 5
Device Id: 25
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c4158d
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SGAD1130BA23

Physical Disk: 5
Enclosure Device ID: 33
Slot Number: 6
Device Id: 26
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c3f4a5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SFVR1130BA23

Physical Disk: 6
Enclosure Device ID: 33
Slot Number: 7
Device Id: 27
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c474e9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7H51130BA23

Physical Disk: 7
Enclosure Device ID: 33
Slot Number: 8
Device Id: 28
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46005
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0QZTV1130BA23

Physical Disk: 8
Enclosure Device ID: 33
Slot Number: 9
Device Id: 29
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c476b5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R7N41130BA23

Physical Disk: 9
Enclosure Device ID: 33
Slot Number: 10
Device Id: 30
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c468f5
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R8721130BA23

Physical Disk: 10
Enclosure Device ID: 33
Slot Number: 11
Device Id: 31
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c46c29
SAS Address(1): 0x0

Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0R81C1130BA23

Physical Disk: 11
Enclosure Device ID: 33
Slot Number: 12
Device Id: 32
Sequence Number: 2
Media Error Count: 0
Other Error Count: 0
Predictive Failure Count: 0
Last Predictive Failure Event Seq Number: 0
Raw Size: 70006MB [0x88bb6b0 Sectors]
Non Coerced Size: 69494MB [0x87bb6b0 Sectors]
Coerced Size: 68664MB [0x861c000 Sectors]
Firmware state: Online
SAS Address(0): 0x5000c50001c40de9
SAS Address(1): 0x0
Connected Port Number: 4(path0)
Inquiry Data: IBM-ESXSST373455SS BA233LQ0SG7H1130BA23

Windows System Information

System Information report written at: 11/14/07 10:35:42
System Name: TPCHA
[System Summary]

Item Value
OS Name Microsoft(R) Windows(R) Server 2003 Enterprise x64 Edition
Version 5.2.3790 Service Pack 2 Build 3790
Other OS Description Not Available
OS Manufacturer Microsoft Corporation
System Name TPCHA
System Manufacturer IBM
System Model IBM 3850 M2 / x3950 M2 -[71414RZ]-
System Type x64-based PC
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
Processor EM64T Family 6 Model 15 Stepping 11 GenuineIntel ~2932 Mhz
BIOS Version/Date IBM -[A3E109TUS-1.00]-, 10/12/2007
SMBIOS Version 2.4
Windows Directory C:\WINDOWS
System Directory C:\WINDOWS\system32
Boot Device \Device\HarddiskVolume1
Locale United States
Hardware Abstraction Layer Version = "5.2.3790.3959 (srv03_sp2_rtm.070216-1710)"
User NameTPCHA\Administrator
Time ZoneEastern Standard Time
Total Physical Memory 261,630.57 MB
Available Physical Memory 247.10 GB
Total Virtual Memory 266.03 GB
Available Virtual Memory 263.10 GB
Page File Space 15.98 GB
Page File L:\pagefile.sys

[Hardware Resources]

[Conflicts/Sharing]

Resource Device
Memory Address 0xF0000000-0xF41FFFFF PCI bus
Memory Address 0xF0000000-0xF41FFFFF PCI standard PCI-to-PCI bridge
Memory Address 0xF0000000-0xF41FFFFF Broadcom BCM5709C NetXtreme II GigE

I/O Port 0x00000000-0x00000CF7 PCI bus
I/O Port 0x00000000-0x00000CF7 Direct memory access controller

I/O Port 0x00004400-0x000044FF PCI bus
I/O Port 0x00004400-0x000044FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xFE000000-0xFE0FFFFF PCI bus
Memory Address 0xFE000000-0xFE0FFFFF PCI bus

Memory Address 0xF4400000-0xF44FFFFF PCI bus
Memory Address 0xF4400000-0xF44FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xF4700000-0xF47FFFFF PCI bus
Memory Address 0xF4700000-0xF47FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

I/O Port 0x00003700-0x000037FF PCI bus
I/O Port 0x00003700-0x000037FF LSI Adapter, SAS RAID-on-Chip, 8-port with 1078 -StorPort

I/O Port 0x00003500-0x000035FF PCI bus
I/O Port 0x00003500-0x000035FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xE6600000-0xE66FFFFF PCI bus
Memory Address 0xE6600000-0xE66FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xE8000000-0xEFFFFFFF PCI bus
Memory Address 0xE8000000-0xEFFFFFFF Standard VGA Graphics Adapter

Memory Address 0xE6300000-0xE63FFFFF PCI bus
Memory Address 0xE6300000-0xE63FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

I/O Port 0x00003800-0x000038FF PCI bus
I/O Port 0x00003800-0x000038FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

IRQ 23 Standard Universal PCI to USB Host Controller
IRQ 23 Standard Enhanced PCI to USB Host Controller

I/O Port 0x00003400-0x000034FF PCI bus
I/O Port 0x00003400-0x000034FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

I/O Port 0x00004600-0x000046FF PCI bus

I/O Port 0x00004600-0x000046FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

I/O Port 0x00003B00-0x00003BFF PCI bus
I/O Port 0x00003B00-0x00003BFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xF4900000-0xF49FFFFF PCI bus
Memory Address 0xF4900000-0xF49FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

I/O Port 0x00004200-0x000042FF PCI bus
I/O Port 0x00004200-0x000042FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xF4200000-0xF42FFFFF PCI bus
Memory Address 0xF4200000-0xF42FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xF4600000-0xF46FFFFF PCI bus
Memory Address 0xF4600000-0xF46FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xE6500000-0xE65FFFFF PCI bus
Memory Address 0xE6500000-0xE65FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

IRQ 122 Standard Universal PCI to USB Host Controller
IRQ 122 Standard Enhanced PCI to USB Host Controller

Memory Address 0xF4500000-0xF45FFFFF PCI bus
Memory Address 0xF4500000-0xF45FFFFF LSI Adapter, SAS RAID-on-Chip, 8-port with 1078 -StorPort

I/O Port 0x00004500-0x000045FF PCI bus
I/O Port 0x00004500-0x000045FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xE2000000-0xE60FFFFF PCI bus
Memory Address 0xE2000000-0xE60FFFFF PCI standard PCI-to-PCI bridge
Memory Address 0xE2000000-0xE60FFFFF Broadcom BCM5709C NetXtreme II GigE

IRQ 17 Broadcom BCM5709C NetXtreme II GigE
IRQ 17 Standard Universal PCI to USB Host Controller

I/O Port 0x00004100-0x000041FF PCI bus
I/O Port 0x00004100-0x000041FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xE6400000-0xE64FFFFF PCI bus
Memory Address 0xE6400000-0xE64FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xF4800000-0xF48FFFFF PCI bus
Memory Address 0xF4800000-0xF48FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xA0000-0xBFFFF PCI bus
Memory Address 0xA0000-0xBFFFF Standard VGA Graphics Adapter

I/O Port 0x00003900-0x000039FF PCI bus
I/O Port 0x00003900-0x000039FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

IRQ 116 Broadcom BCM5709C NetXtreme II GigE
IRQ 116 Standard Universal PCI to USB Host Controller

Memory Address 0xF4300000-0xF43FFFFF PCI bus
Memory Address 0xF4300000-0xF43FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

I/O Port 0x00004300-0x000043FF PCI bus
I/O Port 0x00004300-0x000043FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xE6200000-0xE62FFFFF PCI bus
Memory Address 0xE6200000-0xE62FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

I/O Port 0x00003A00-0x00003AFF PCI bus
I/O Port 0x00003A00-0x00003AFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

Memory Address 0xE6100000-0xE61FFFFF PCI bus
Memory Address 0xE6100000-0xE61FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

I/O Port 0x00003600-0x000036FF PCI bus
I/O Port 0x00003600-0x000036FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller

[DMA]

Resource Device Status
Channel 4 Direct memory access controller OK

[Forced Hardware]

Device PNP Device ID

[I/O]

Resource Device Status
0x00000000-0x00000CF7 PCI bus OK
0x00000000-0x00000CF7 Direct memory access controller OK
0x00000D00-0x000033FF PCI bus OK
0x00003000-0x0000301F Standard Universal PCI to USB Host Controller OK
0x00003100-0x0000311F Standard Universal PCI to USB Host Controller OK
0x00003200-0x0000321F Standard Universal PCI to USB Host Controller OK
0x00003300-0x0000331F Standard Universal PCI to USB Host Controller OK
0x00002000-0x000020FF Standard VGA Graphics Adapter OK
0x000003B0-0x000003BB Standard VGA Graphics Adapter OK
0x000003C0-0x000003DF Standard VGA Graphics Adapter OK
0x000003F8-0x000003FF Communications Port (COM1) OK
0x00000020-0x00000021 Advanced programmable interrupt controller OK
0x000000A0-0x000000A1 Advanced programmable interrupt controller OK
0x00000080-0x0000008F Direct memory access controller OK
0x000000C0-0x000000DF Direct memory access controller OK
0x00000040-0x00000043 System timer OK
0x00000070-0x00000073 System CMOS/real time clock OK
0x00000061-0x00000061 System speaker OK
0x000000F0-0x000000FF Numeric data processor OK
0x00000024-0x00000025 Motherboard resources OK
0x00000028-0x00000029 Motherboard resources OK
0x0000002C-0x0000002D Motherboard resources OK
0x00000030-0x00000031 Motherboard resources OK
0x00000034-0x00000035 Motherboard resources OK
0x00000038-0x00000039 Motherboard resources OK
0x0000003C-0x0000003D Motherboard resources OK
0x0000004E-0x0000004F Motherboard resources OK
0x00000050-0x00000052 Motherboard resources OK
0x00000060-0x00000060 Motherboard resources OK
0x00000064-0x00000064 Motherboard resources OK
0x00000074-0x00000077 Motherboard resources OK
0x00000090-0x0000009F Motherboard resources OK
0x000000A4-0x000000A5 Motherboard resources OK
0x000000A8-0x000000A9 Motherboard resources OK
0x000000AC-0x000000AD Motherboard resources OK
0x000000B0-0x000000B5 Motherboard resources OK
0x000000B8-0x000000B9 Motherboard resources OK
0x000000BC-0x000000BD Motherboard resources OK

0x00000400-0x0000043F Motherboard resources OK
0x00000440-0x0000045F Motherboard resources OK
0x000004D0-0x000004D1 Motherboard resources OK
0x00000540-0x0000055F Motherboard resources OK
0x00000560-0x0000057F Motherboard resources OK
0x00000580-0x000005FF Motherboard resources OK
0x00000C00-0x00000CA7 Motherboard resources OK
0x00000CB0-0x00000CDE Motherboard resources OK
0x00000CA8-0x00000CA8 Motherboard resources OK
0x00000CAC-0x00000CAC Motherboard resources OK
0x00000700-0x0000070F Standard Dual Channel PCI IDE Controller OK
0x000001F0-0x000001F7 Primary IDE Channel OK
0x000003F6-0x000003F6 Primary IDE Channel OK
0x00000170-0x00000177 Secondary IDE Channel OK
0x00000376-0x00000376 Secondary IDE Channel OK
0x00003700-0x000037FF PCI bus OK
0x00003700-0x000037FF LSI Adapter, SAS RAID-on-Chip, 8-port with 1078 -StorPort OK
0x00003800-0x000038FF PCI bus OK
0x00003800-0x000038FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00003900-0x000039FF PCI bus OK
0x00003900-0x000039FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00003A00-0x00003AFF PCI bus OK
0x00003A00-0x00003AFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00003B00-0x00003BFF PCI bus OK
0x00003B00-0x00003BFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00003400-0x000034FF PCI bus OK
0x00003400-0x000034FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00003500-0x000035FF PCI bus OK
0x00003500-0x000035FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00003600-0x000036FF PCI bus OK
0x00003600-0x000036FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00003C00-0x000040FF PCI bus OK
0x00003D00-0x00003D1F Standard Universal PCI to USB Host Controller OK
0x00003E00-0x00003E1F Standard Universal PCI to USB Host Controller OK
0x00003F00-0x00003F1F Standard Universal PCI to USB Host Controller OK
0x00004000-0x0000401F Standard Universal PCI to USB Host Controller OK
0x00004400-0x000044FF PCI bus OK
0x00004400-0x000044FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00004500-0x000045FF PCI bus OK
0x00004500-0x000045FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00004600-0x000046FF PCI bus OK
0x00004600-0x000046FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00004100-0x000041FF PCI bus OK
0x00004100-0x000041FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00004200-0x000042FF PCI bus OK
0x00004200-0x000042FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0x00004300-0x000043FF PCI bus OK
0x00004300-0x000043FF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK

[IRQs]

Resource Device Status
IRQ 9 Microsoft ACPI-Compliant System OK
IRQ 16 Broadcom BCM5709C NetXtreme II GigE OK
IRQ 17 Broadcom BCM5709C NetXtreme II GigE OK
IRQ 17 Standard Universal PCI to USB Host Controller OK
IRQ 23 Standard Universal PCI to USB Host Controller OK
IRQ 23 Standard Enhanced PCI to USB Host Controller OK
IRQ 18 Standard Universal PCI to USB Host Controller OK
IRQ 19 Standard Universal PCI to USB Host Controller OK
IRQ 4 Communications Port (COM1) OK
IRQ 0 System timer OK
IRQ 8 System CMOS/real time clock OK
IRQ 13 Numeric data processor OK
IRQ 14 Primary IDE Channel OK
IRQ 46 LSI Adapter, SAS RAID-on-Chip, 8-port with 1078 -StorPort OK
IRQ 79 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 80 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 81 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 82 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 43 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 44 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 45 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 115 Broadcom BCM5709C NetXtreme II GigE OK
IRQ 116 Broadcom BCM5709C NetXtreme II GigE OK
IRQ 116 Standard Universal PCI to USB Host Controller OK
IRQ 122 Standard Universal PCI to USB Host Controller OK
IRQ 122 Standard Enhanced PCI to USB Host Controller OK
IRQ 117 Standard Universal PCI to USB Host Controller OK
IRQ 118 Standard Universal PCI to USB Host Controller OK
IRQ 175 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 176 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 177 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 139 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 140 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
IRQ 141 IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK

[Memory]

Resource Device Status
0xA0000-0xBFFFF PCI bus OK
0xA0000-0xBFFFF Standard VGA Graphics Adapter OK
0xE8000000-0xEFFFFFFF PCI bus OK
0xE8000000-0xEFFFFFFF Standard VGA Graphics Adapter OK
0xF0000000-0xF41FFFFF PCI bus OK
0xF0000000-0xF41FFFFF PCI standard PCI-to-PCI bridge OK
0xF0000000-0xF41FFFFF Broadcom BCM5709C NetXtreme II GigE OK
0xF2000000-0xF3FFFFFF Broadcom BCM5709C NetXtreme II GigE OK
0xF4100000-0xF41003FF Standard Enhanced PCI to USB Host Controller OK
0xF4020000-0xF402FFFF Standard VGA Graphics Adapter OK
0xFDE86000-0xFDF05FFF Motherboard resources OK
0xFDA00000-0xFDBFFFFF Motherboard resources OK
0x0400-0x04FF System board OK
0xFED40000-0xFED44FFF Motherboard resources OK
0x100000-0xCFFFFFFF Memory Module OK
0xF4500000-0xF45FFFFF PCI bus OK
0xF4500000-0xF45FFFFF LSI Adapter, SAS RAID-on-Chip, 8-port with 1078 -StorPort OK
0xF4550000-0xF4553FFF LSI Adapter, SAS RAID-on-Chip, 8-port with 1078 -StorPort OK
0xF4600000-0xF46FFFFF PCI bus OK
0xF4600000-0xF46FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4640000-0xF467FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4700000-0xF47FFFFF PCI bus OK
0xF4700000-0xF47FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4740000-0xF477FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4800000-0xF48FFFFF PCI bus OK
0xF4800000-0xF48FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4840000-0xF487FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4900000-0xF49FFFFF PCI bus OK
0xF4900000-0xF49FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK

0xF4940000-0xF497FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4200000-0xF42FFFFF PCI bus OK
0xF4200000-0xF42FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4240000-0xF427FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4300000-0xF43FFFFF PCI bus OK
0xF4300000-0xF43FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4340000-0xF437FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4400000-0xF44FFFFF PCI bus OK
0xF4400000-0xF44FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xF4440000-0xF447FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE2000000-0xE60FFFFF PCI bus OK
0xE2000000-0xE60FFFFF PCI standard PCI-to-PCI bridge OK
0xE2000000-0xE60FFFFF Broadcom BCM5709C NetXtreme II GigE OK
0xE4000000-0xE5FFFFFF Broadcom BCM5709C NetXtreme II GigE OK
0xE6000000-0xE60003FF Standard Enhanced PCI to USB Host Controller OK
0xFE000000-0xFE0FFFFF PCI bus OK
0xFE000000-0xFE0FFFFF PCI bus OK
0xE6400000-0xE64FFFFF PCI bus OK
0xE6400000-0xE64FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6440000-0xE647FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6500000-0xE65FFFFF PCI bus OK
0xE6500000-0xE65FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6540000-0xE657FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6600000-0xE66FFFFF PCI bus OK
0xE6600000-0xE66FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6640000-0xE667FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6100000-0xE61FFFFF PCI bus OK
0xE6100000-0xE61FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6140000-0xE617FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6200000-0xE62FFFFF PCI bus OK
0xE6200000-0xE62FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6240000-0xE627FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6300000-0xE63FFFFF PCI bus OK
0xE6300000-0xE63FFFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK
0xE6340000-0xE637FFFF IBM SystemX MegaRAID SAS 8880EM2 RAID Controller OK

[Components]

[Multimedia]

[Audio Codecs]

CODEC Manufacturer Description Status File Version Size Creation Date
c:\windows\system32\msadp32.acm Microsoft Corporation OK C:\WINDOWS\system32\MSADP32.ACM 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 23.50 KB (24,064 bytes) 3/25/2005 7:00 AM
c:\windows\system32\tssoft32.acm DSP GROUP, INC. OK C:\WINDOWS\system32\TSSOFT32.ACM 1.01 13.50 KB (13,824 bytes) 3/25/2005 7:00 AM
c:\windows\system32\msgsm32.acm Microsoft Corporation OK C:\WINDOWS\system32\MSGSM32.ACM 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 34.50 KB (35,328 bytes) 3/25/2005 7:00 AM
c:\windows\system32\msg711.acm Microsoft Corporation OK C:\WINDOWS\system32\MSG711.ACM 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 13.50 KB (13,824 bytes) 3/25/2005 7:00 AM
c:\windows\system32\imaadp32.acm Microsoft Corporation OK C:\WINDOWS\system32\IMAADP32.ACM 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 24.00 KB (24,576 bytes) 3/25/2005 7:00 AM

[Video Codecs]

CODEC Manufacturer Description Status File Version Size Creation Date
c:\windows\system32\iyuv_32.dll Microsoft Corporation OK C:\WINDOWS\system32\IYUV_32.DLL 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 52.50 KB (53,760 bytes) 3/24/2005 12:19 PM
c:\windows\system32\msrle32.dll Microsoft Corporation OK C:\WINDOWS\system32\MSRLE32.DLL 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 15.50 KB (15,872 bytes) 3/25/2005 7:00 AM
c:\windows\system32\msvidc32.dll Microsoft Corporation OK C:\WINDOWS\system32\MSVIDC32.DLL 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 43.00 KB (44,032 bytes) 3/25/2005 7:00 AM
c:\windows\system32\msyuv.dll Microsoft Corporation OK C:\WINDOWS\system32\MSYUV.DLL 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 21.00 KB (21,504 bytes) 3/24/2005 12:21 PM
c:\windows\system32\tsbyuv.dll Microsoft Corporation OK C:\WINDOWS\system32\TSBYUV.DLL 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 12.50 KB (12,800 bytes) 3/24/2005 12:34 PM

[CD-ROM]

Item Value
Drive K:
Description CD-ROM Drive
Media Loaded No
Media Type CD-ROM
Name HL-DT-ST RW/DVD GCC-T10N
Manufacturer (Standard CD-ROM drives)
Status OK
Transfer Rate Not Available
SCSI Target ID 0
PNP Device ID IDE\CDROMHL-DT-ST_RW/DVD_GCC-T10N_____1.00____\5&E769355&0&0.0.0
Driver c:\windows\system32\drivers\cdrom.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 75.50 KB (77,312 bytes), 3/25/2005 7:00 AM)

[Sound Device]

Item Value

[Display]

Item Value
Name Standard VGA Graphics Adapter
PNP Device ID PCI\VEN_1002&DEV_515E&SUBSYS_03191014&REV_02\4&18E510B5&0&00F0
Adapter Type ATI ES1000, (Standard display types) compatible
Adapter Description Standard VGA Graphics Adapter
Adapter RAM 16.00 MB (16,777,216 bytes)
Installed Drivers vga.dll,framebuf.dll,vga256,vga64k

Driver Version 5.2.3790.1830
INF File display.inf (vga section)
Color Planes 1
Color Table Entries 4294967296
Resolution 1024 x 768 x 1 hertz
Bits/Pixel 32
Memory Address 0xE8000000-0xEFFFFFFF
I/O Port 0x00002000-0x000020FF
Memory Address 0xF4020000-0xF402FFFF
I/O Port 0x000003B0-0x000003BB
I/O Port 0x000003C0-0x000003DF
Memory Address 0xA0000-0xBFFFF
Driver c:\windows\system32\drivers\vgapnp.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 33.00 KB (33,792 bytes), 7/11/2007 6:02 AM)

[Infrared]

Item Value

[Input]

[Keyboard]

Item Value
Description USB Human Interface Device
Name Enhanced (101- or 102-key)
Layout 00000409
PNP Device ID USB\VID_0624&PID_0200&MI_00\6&3675DEE2&0&0000
Number of Function Keys 12
Driver c:\windows\system32\drivers\hidusb.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 18.50 KB (18,944 bytes), 3/25/2005 7:00 AM)

Description USB Human Interface Device
Name Enhanced (101- or 102-key)
Layout 00000409
PNP Device ID USB\VID_04B3&PID_4001&MI_00\6&B4B3E6A&0&0000
Number of Function Keys 12
Driver c:\windows\system32\drivers\hidusb.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 18.50 KB (18,944 bytes), 3/25/2005 7:00 AM)

Description USB Human Interface Device
Name Enhanced (101- or 102-key)
Layout 00000409
PNP Device ID USB\VID_0624&PID_0200&MI_00\6&3675DEE2&0&0000
Number of Function Keys 12
Driver c:\windows\system32\drivers\hidusb.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 18.50 KB (18,944 bytes), 3/25/2005 7:00 AM)

Description USB Human Interface Device
Name Enhanced (101- or 102-key)
Layout 00000409
PNP Device ID USB\VID_04B3&PID_4001&MI_00\6&B4B3E6A&0&0000
Number of Function Keys 12
Driver c:\windows\system32\drivers\hidusb.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 18.50 KB (18,944 bytes), 3/25/2005 7:00 AM)

[Pointing Device]

Item Value
Hardware Type HID-compliant mouse
Number of Buttons 5
Status OK
PNP Device ID HID\VID_0624&PID_0200&MI_01&COL01\7&5A1E937&0&0000
Power Management Supported No
Double Click Threshold 6
Handedness Right Handed Operation
Driver c:\windows\system32\drivers\mouhid.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 19.00 KB (19,456 bytes), 3/24/2005 12:21 PM)

Hardware Type USB Human Interface Device
Number of Buttons 5
Status OK
PNP Device ID USB\VID_04B3&PID_4001&MI_01\6&B4B3E6A&0&0001
Power Management Supported No
Double Click Threshold 6
Handedness Right Handed Operation
Driver c:\windows\system32\drivers\hidusb.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 18.50 KB (18,944 bytes), 3/25/2005 7:00 AM)

Hardware Type HID-compliant mouse
Number of Buttons 5
Status OK
PNP Device ID HID\VID_0624&PID_0200&MI_01&COL01\7&12889A6&0&0000
Power Management Supported No
Double Click Threshold 6
Handedness Right Handed Operation
Driver c:\windows\system32\drivers\mouhid.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 19.00 KB (19,456 bytes), 3/24/2005 12:21 PM)

Hardware Type USB Human Interface Device
Number of Buttons 5
Status OK
PNP Device ID USB\VID_04B3&PID_4001&MI_01\6&B4B3E6A&0&0001
Power Management Supported No
Double Click Threshold 6
Handedness Right Handed Operation
Driver c:\windows\system32\drivers\hidusb.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 18.50 KB (18,944 bytes), 3/25/2005 7:00 AM)

[Modem]

Item Value

[Network]

[Adapter]

Item Value
Name [00000001] RAS Async Adapter
Adapter Type Not Available
Product Type RAS Async Adapter
Installed Yes
PNP Device ID Not Available
Last Reset 11/14/2007 8:37 AM
Index 1
Service Name AsyncMac
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available

Name [00000002] WAN Miniport (L2TP)
Adapter Type Not Available
Product Type WAN Miniport (L2TP)
Installed Yes
PNP Device ID ROOT\MS_L2TPMINIPORT\0000
Last Reset 11/14/2007 8:37 AM
Index 2
Service Name Rasl2tp
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available

Driver c:\windows\system32\drivers\rasl2tp.sys (5.2.3790.3959 (srv03_sp2_rtm.070216-1710), 132.00 KB (135,168 bytes), 3/25/2005 7:00 AM)

Name [00000003] WAN Miniport (PPTP)
Adapter Type Wide Area Network (WAN)
Product Type WAN Miniport (PPTP)
Installed Yes
PNP Device ID ROOT\MS_PPTPMINIPORT\0000
Last Reset 11/14/2007 8:37 AM
Index 3
Service Name PptpMiniport
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address 50:50:54:50:30:30
Driver c:\windows\system32\drivers\raspptp.sys (5.2.3790.3959 (srv03_sp2_rtm.070216-1710), 117.50 KB (120,320 bytes), 3/25/2005 7:00 AM)

Name [00000004] WAN Miniport (PPPOE)
Adapter Type Wide Area Network (WAN)
Product Type WAN Miniport (PPPOE)
Installed Yes
PNP Device ID ROOT\MS_PPPOEMINIPORT\0000
Last Reset 11/14/2007 8:37 AM
Index 4
Service Name RasPppoe
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address 33:50:6F:45:30:30
Driver c:\windows\system32\drivers\raspppoe.sys (5.2.3790.3959 (srv03_sp2_rtm.070216-1710), 67.50 KB (69,120 bytes), 3/25/2005 7:00 AM)

Name [00000005] Direct Parallel
Adapter Type Not Available
Product Type Direct Parallel
Installed Yes
PNP Device ID ROOT\MS_PTIMINIPORT\0000
Last Reset 11/14/2007 8:37 AM
Index 5
Service Name Raspti
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available
Driver c:\windows\system32\drivers\raspti.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 30.50 KB (31,232 bytes), 3/25/2005 7:00 AM)

Name [00000006] WAN Miniport (IP)
Adapter Type Not Available
Product Type WAN Miniport (IP)
Installed Yes
PNP Device ID ROOT\MS_NDISWANIP\0000
Last Reset 11/14/2007 8:37 AM
Index 6
Service Name NdisWan
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available
Driver c:\windows\system32\drivers\ndiswan.sys (5.2.3790.3959 (srv03_sp2_rtm.070216-1710), 157.50 KB (161,280 bytes), 3/25/2005 7:00 AM)

Name [00000007] Intel(R) PRO/1000 PT Dual Port Server Adapter
Adapter Type Not Available
Product Type Intel(R) PRO/1000 PT Dual Port Server Adapter
Installed Yes
PNP Device ID Not Available
Last Reset 11/14/2007 8:37 AM
Index 7
Service Name e1express
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available

Name [00000008] Intel(R) PRO/1000 PT Dual Port Server Adapter
Adapter Type Not Available
Product Type Intel(R) PRO/1000 PT Dual Port Server Adapter
Installed Yes
PNP Device ID Not Available
Last Reset 11/14/2007 8:37 AM
Index 8
Service Name e1express
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled Yes
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available

Name [00000009] Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Adapter Type Not Available
Product Type Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Installed Yes
PNP Device ID Not Available
Last Reset 11/14/2007 8:37 AM
Index 9
Service Name l2nd
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available

Name [00000010] Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Adapter Type Not Available
Product Type Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Installed Yes
PNP Device ID Not Available
Last Reset 11/14/2007 8:37 AM
Index 10
Service Name l2nd
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled Yes
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available

Name [00000011] Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Adapter Type Not Available
Product Type Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Installed Yes
PNP Device ID Not Available
Last Reset 11/14/2007 8:37 AM
Index 11
Service Name l2nd
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available

Name [00000012] Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Adapter Type Not Available
Product Type Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Installed Yes
PNP Device ID Not Available
Last Reset 11/14/2007 8:37 AM
Index 12
Service Name l2nd
IP Address Not Available
IP Subnet Not Available
Default IP Gateway Not Available
DHCP Enabled Yes
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address Not Available

Name [00000013] Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Adapter Type Ethernet 802.3
Product Type Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Installed Yes
PNP Device ID
B06BDRV\L2ND&PCI_163914E4&SUBSYS_037C1014&REV_01\5&158780AB&0&20050200
Last Reset 11/14/2007 8:37 AM
Index 13
Service Name l2nd
IP Address 192.168.30.202
IP Subnet 255.255.255.0
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address 00:1A:64:35:10:CC
Driver c:\windows\system32\drivers\bxnd52a.sys (3.7.4.0 built by: WinDDK, 63.50 KB (65,024 bytes), 7/24/2007 2:38 PM)

Name [00000014] Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Adapter Type Ethernet 802.3
Product Type Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Installed Yes
PNP Device ID
B06BDRV\L2ND&PCI_163914E4&SUBSYS_037C1014&REV_01\5&38892308&0&20050200
Last Reset 11/14/2007 8:37 AM
Index 14
Service Name l2nd
IP Address 0.0.0.0
IP Subnet 0.0.0.0
Default IP Gateway Not Available
DHCP Enabled No
DHCP Server Not Available
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address 00:1A:64:35:10:CE
Driver c:\windows\system32\drivers\bxnd52a.sys (3.7.4.0 built by: WinDDK, 63.50 KB (65,024 bytes), 7/24/2007 2:38 PM)

Name [00000015] Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Adapter Type Ethernet 802.3
Product Type Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Installed Yes
PNP Device ID
B06BDRV\L2ND&PCI_163914E4&SUBSYS_037C1014&REV_01\5&20CCE83A&0&20053100
Last Reset 11/14/2007 8:37 AM
Index 15
Service Name l2nd
IP Address 0.0.0.0
IP Subnet 0.0.0.0
Default IP Gateway Not Available
DHCP Enabled Yes
DHCP Server
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address 00:1A:64:36:01:12
Driver c:\windows\system32\drivers\bxnd52a.sys (3.7.4.0 built by: WinDDK, 63.50 KB (65,024 bytes), 7/24/2007 2:38 PM)

Name [00000016] Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Adapter Type Ethernet 802.3
Product Type Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client)
Installed Yes
PNP Device ID
B06BDRV\L2ND&PCI_163914E4&SUBSYS_037C1014&REV_01\5&833C090&0&20053100
Last Reset 11/14/2007 8:37 AM
Index 16
Service Name l2nd
IP Address 0.0.0.0
IP Subnet 0.0.0.0
Default IP Gateway Not Available
DHCP Enabled Yes
DHCP Server
DHCP Lease Expires Not Available
DHCP Lease Obtained Not Available
MAC Address 00:1A:64:36:01:10
Driver c:\windows\system32\drivers\bxnd52a.sys (3.7.4.0 built by: WinDDK, 63.50 KB (65,024 bytes), 7/24/2007 2:38 PM)

[Protocol]

Item Value
Name MSAFD Tcpip [TCP/IP]
Connectionless Service No
Guarantees Delivery Yes
Guarantees Sequencing Yes
Maximum Address Size 16 bytes
Maximum Message Size 0 bytes
Message Oriented No
Minimum Address Size 16 bytes
Pseudo Stream Oriented No
Supports Broadcasting No
Supports Connect Data No
Supports Disconnect Data No
Supports Encryption No
Supports Expedited Data Yes
Supports Graceful Closing Yes
Supports Guaranteed Bandwidth No
Supports Multicasting No

Name MSAFD Tcpip [UDP/IP]
Connectionless Service Yes

Guarantees Delivery No
Guarantees Sequencing No
Maximum Address Size 16 bytes
Maximum Message Size 63.93 KB (65,467 bytes)
Message Oriented Yes
Minimum Address Size 16 bytes
Pseudo Stream Oriented No
Supports Broadcasting Yes
Supports Connect Data No
Supports Disconnect Data No
Supports Encryption No
Supports Expedited Data No
Supports Graceful Closing No
Supports Guaranteed Bandwidth No
Supports Multicasting Yes

Name RSVP UDP Service Provider
Connectionless Service Yes
Guarantees Delivery No
Guarantees Sequencing No
Maximum Address Size 16 bytes
Maximum Message Size 63.93 KB (65,467 bytes)
Message Oriented Yes
Minimum Address Size 16 bytes
Pseudo Stream Oriented No
Supports Broadcasting Yes
Supports Connect Data No
Supports Disconnect Data No
Supports Encryption Yes
Supports Expedited Data No
Supports Graceful Closing No
Supports Guaranteed Bandwidth No
Supports Multicasting Yes

Name RSVP TCP Service Provider
Connectionless Service No
Guarantees Delivery Yes
Guarantees Sequencing Yes
Maximum Address Size 16 bytes
Maximum Message Size 0 bytes
Message Oriented No
Minimum Address Size 16 bytes
Pseudo Stream Oriented No
Supports Broadcasting No
Supports Connect Data No
Supports Disconnect Data No
Supports Encryption Yes
Supports Expedited Data Yes
Supports Graceful Closing Yes
Supports Guaranteed Bandwidth No
Supports Multicasting No

[WinSock]

Item Value
File c:\windows\system32\wsock32.dll
Size 24.50 KB (25,088 bytes)
Version 5.2.3790.1830 (srv03_sp1_rtm.050324-1447)

[Ports]

[Serial]

Item Value
Name Communications Port (COM1)
Status OK
PNP Device ID ACPI\PNP0501\1
Maximum Input Buffer Size 0
Maximum Output Buffer Size No
Settable Baud Rate Yes
Settable Data Bits Yes
Settable Flow Control Yes
Settable Parity Yes
Settable Parity Check Yes
Settable Stop Bits Yes
Settable RLSD Yes
Supports RLSD Yes
Supports 16 Bit Mode No
Supports Special Characters No
Baud Rate 9600
Bits/Byte 8
Stop Bits 1
Parity None
Busy No
Abort Read/Write on Error No
Binary Mode Enabled Yes
Continue XMit on XOff No
CTS Outflow Control No
Discard NULL Bytes No
DSR Outflow Control 0
DSR Sensitivity 0
DTR Flow Control Type Enable
EOF Character 0
Error Replace Character 0
Error Replacement Enabled No
Event Character 0
Parity Check Enabled No
RTS Flow Control Type Enable
XOff Character 19
XOffXMit Threshold 512
XOn Character 17
XOnXMit Threshold 2048
XOnXOff InFlow Control 0
XOnXOff OutFlow Control 0
I/O Port 0x000003F8-0x000003FF
IRQ Channel IRQ 4
Driver c:\windows\system32\drivers\serial.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 118.50 KB (121,344 bytes), 3/25/2005 7:00 AM)

[Parallel]

Item Value

[Storage]

[Drives]

Item Value
Drive C:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 36.00 GB (38,658,781,184 bytes)
Free Space 15.12 GB (16,238,469,120 bytes)
Volume Name
Volume Serial Number ECF4C64A

Drive D:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 31.05 GB (33,337,057,280 bytes)
Free Space 26.22 GB (28,149,780,480 bytes)
Volume Name Run Results
Volume Serial Number 4039026B

Drive E:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 110.15 GB (118,276,042,752 bytes)
Volume Name New Volume
Volume Serial Number A42B4620

Drive F:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 69.46 GB (74,583,920,640 bytes)
Volume Name New Volume
Volume Serial Number 18380ACA

Drive G:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 71.53 GB (76,803,010,560 bytes)
Volume Name New Volume
Volume Serial Number E04CB7BC

Drive H:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 71.53 GB (76,805,640,192 bytes)
Volume Name New Volume
Volume Serial Number 60569953

Drive I:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 71.39 GB (76,650,762,240 bytes)
Volume Name New Volume
Volume Serial Number A8627B91

Drive J:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 71.27 GB (76,522,569,728 bytes)
Volume Name New Volume
Volume Serial Number 3C6E00EB

Drive K:
Description CD-ROM Disc

Drive L:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 68.36 GB (73,402,363,904 bytes)
Free Space 52.31 GB (56,169,402,368 bytes)
Volume Name Pagefile
Volume Serial Number F47F6996

Drive M:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 68.36 GB (73,402,363,904 bytes)
Free Space 63.00 GB (67,642,789,888 bytes)
Volume Name Backup
Volume Serial Number D84BA546

Drive N:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 100.01 GB (107,380,994,048 bytes)
Free Space 99.91 GB (107,276,357,632 bytes)
Volume Name RF_Flatfiles
Volume Serial Number 646125A8

Drive O:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 71.27 GB (76,526,727,168 bytes)
Volume Name New Volume
Volume Serial Number 801B0567

Drive P:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 71.27 GB (76,527,886,336 bytes)
Volume Name New Volume
Volume Serial Number F82626B5

Drive Q:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 112.23 GB (120,502,226,944 bytes)
Volume Name New Volume
Volume Serial Number 44323AC8

Drive R:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 112.23 GB (120,502,366,208 bytes)
Volume Name New Volume
Volume Serial Number 343DFBDD

Drive S:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 112.23 GB (120,502,366,208 bytes)
Volume Name New Volume
Volume Serial Number 44493E22

Drive T:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 112.23 GB (120,502,366,208 bytes)
Volume Name New Volume
Volume Serial Number 0C5A90C0

Drive U:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 112.23 GB (120,502,366,208 bytes)
Volume Name New Volume
Volume Serial Number E4661F9B

Drive V:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 112.23 GB (120,502,366,208 bytes)
Volume Name New Volume
Volume Serial Number 2C7153CC

Drive W:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 112.23 GB (120,502,366,208 bytes)
Volume Name New Volume
Volume Serial Number EC820DCA

Drive X:
Description Local Fixed Disk
Compressed No
File System NTFS
Size 160.00 GB (171,801,387,008 bytes)
Free Space 112.23 GB (120,501,395,456 bytes)
Volume Name New Volume
Volume Serial Number 948C9FCC

[Disks]

Item Value
Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 5
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #7, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 5
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #8, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 13
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #22, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 13
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #23, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 8
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #13, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 8
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #14, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive

Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 12
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #20, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 12
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #21, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 4
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #5, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 4
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #6, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 11
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #18, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 11
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #19, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 15
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #26, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)

Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 15
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #27, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 6
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #9, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 6
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #10, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 3
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #3, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 3
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #4, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 9
SCSI Target ID 0
Sectors/Track 63
Size 402.32 GB (431,991,705,600 bytes)
Total Cylinders 52,520
Total Sectors 843,733,800
Total Tracks 13,392,600
Tracks/Cylinder 255
Partition Disk #15, Partition #0
Partition Size 402.32 GB (431,983,480,320 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 7
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 7
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #11, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device

Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 7
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 7
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #12, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 10
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #16, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 5
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 10
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #17, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 14
SCSI Target ID 0
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #24, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM MegaRAID 8880EM2 SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 4
SCSI Bus 1
SCSI Logical Unit 0
SCSI Port 14
SCSI Target ID 1
Sectors/Track 63
Size 804.65 GB (863,991,636,480 bytes)
Total Cylinders 105,041
Total Sectors 1,687,483,665
Total Tracks 26,785,455
Tracks/Cylinder 255
Partition Disk #25, Partition #0
Partition Size 804.65 GB (863,983,411,200 bytes)
Partition Starting Offset 8,225,280 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM-ESXS MAY2073RC SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 1
SCSI Bus 0
SCSI Logical Unit 0
SCSI Port 2
SCSI Target ID 10
Sectors/Track 63
Size 68.36 GB (73,402,398,720 bytes)
Total Cylinders 8,924
Total Sectors 143,364,060
Total Tracks 2,275,620
Tracks/Cylinder 255
Partition Disk #1, Partition #0
Partition Size 68.36 GB (73,402,366,464 bytes)
Partition Starting Offset 32,256 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model IBM-ESXS ST973401SS SCSI Disk Device
Bytes/Sector 512
Media Loaded Yes
Media Type Fixed hard disk
Partitions 1
SCSI Bus 0
SCSI Logical Unit 0
SCSI Port 2
SCSI Target ID 11
Sectors/Track 63
Size 68.36 GB (73,402,398,720 bytes)
Total Cylinders 8,924
Total Sectors 143,364,060
Total Tracks 2,275,620
Tracks/Cylinder 255
Partition Disk #2, Partition #0
Partition Size 68.36 GB (73,402,366,464 bytes)
Partition Starting Offset 32,256 bytes

Description Disk drive
Manufacturer (Standard disk drives)
Model LSILOGIC Logical Volume SCSI Disk Device
Bytes/Sector 512

Media Loaded Yes
Media Type Fixed hard disk
Partitions 2
SCSI Bus 0
SCSI Logical Unit 0
SCSI Port 2
SCSI Target ID 8
Sectors/Track 63
Size 67.05 GB (71,995,875,840 bytes)
Total Cylinders 8,753
Total Sectors 140,616,945
Total Tracks 2,232,015
Tracks/Cylinder 255
Partition Disk #0, Partition #0
Partition Size 36.00 GB (38,658,783,744 bytes)
Partition Starting Offset 32,256 bytes
Partition Disk #0, Partition #1
Partition Size 31.05 GB (33,337,059,840 bytes)
Partition Starting Offset 38,658,816,000 bytes

[SCSI]

Item Value
Name LSI Adapter, SAS RAID-on-Chip, 8-port with 1078 -StorPort
Manufacturer LSI Corporation
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0062&SUBSYS_10001000&REV_03\4&2BCCA3F5&0&0000
Memory Address 0xF4550000-0xF4553FFF
I/O Port 0x00003700-0x000037FF
Memory Address 0xF4500000-0xF45FFFFF
IRQ Channel IRQ 46
Driver c:\windows\system32\drivers\lsi_sas.sys (1.25.10.07 built by: WinDDK, 120.50 KB (123,392 bytes), 7/11/2007 5:43 AM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&1DD35BBD&0&0000
Memory Address 0xF4600000-0xF46FFFFF
I/O Port 0x00003800-0x000038FF
Memory Address 0xF4640000-0xF467FFFF
IRQ Channel IRQ 79
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&15BF98E9&0&0000
Memory Address 0xF4700000-0xF47FFFFF
I/O Port 0x00003900-0x000039FF
Memory Address 0xF4740000-0xF477FFFF
IRQ Channel IRQ 80
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&98F6FDA&0&0000
Memory Address 0xF4800000-0xF48FFFFF
I/O Port 0x00003A00-0x00003AFF
Memory Address 0xF4840000-0xF487FFFF
IRQ Channel IRQ 81
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&394070D2&0&0000
Memory Address 0xF4900000-0xF49FFFFF
I/O Port 0x00003B00-0x00003BFF
Memory Address 0xF4940000-0xF497FFFF
IRQ Channel IRQ 82
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&1DD220B0&0&0000
Memory Address 0xF4200000-0xF42FFFFF
I/O Port 0x00003400-0x000034FF
Memory Address 0xF4240000-0xF427FFFF
IRQ Channel IRQ 43
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&167C2C2F&0&0000
Memory Address 0xF4300000-0xF43FFFFF
I/O Port 0x00003500-0x000035FF
Memory Address 0xF4340000-0xF437FFFF
IRQ Channel IRQ 44
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&2657B4F5&0&0000
Memory Address 0xF4400000-0xF44FFFFF
I/O Port 0x00003600-0x000036FF
Memory Address 0xF4440000-0xF447FFFF
IRQ Channel IRQ 45
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&3ED7AD6&0&0000
Memory Address 0xE6400000-0xE64FFFFF
I/O Port 0x00004400-0x000044FF
Memory Address 0xE6440000-0xE647FFFF
IRQ Channel IRQ 175
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&13C9039C&0&0000
Memory Address 0xE6500000-0xE65FFFFF
I/O Port 0x00004500-0x000045FF
Memory Address 0xE6540000-0xE657FFFF

IRQ Channel IRQ 176
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&2111E872&0&000
0
Memory Address 0xE6600000-0xE66FFFFF
I/O Port 0x00004600-0x000046FF
Memory Address 0xE6640000-0xE667FFFF
IRQ Channel IRQ 177
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&A41217E&0&0000
Memory Address 0xE6100000-0xE61FFFFF
I/O Port 0x00004100-0x000041FF
Memory Address 0xE6140000-0xE617FFFF
IRQ Channel IRQ 139
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&36CC30BC&0&00
00
Memory Address 0xE6200000-0xE62FFFFF
I/O Port 0x00004200-0x000042FF
Memory Address 0xE6240000-0xE627FFFF
IRQ Channel IRQ 140
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

Name IBM SystemX MegaRAID SAS 8880EM2 RAID Controller
Manufacturer LSI Corp.,
Status OK
PNP Device ID
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&10B57C39&0&000
0
Memory Address 0xE6300000-0xE63FFFFF
I/O Port 0x00004300-0x000043FF
Memory Address 0xE6340000-0xE637FFFF
IRQ Channel IRQ 141
Driver c:\windows\system32\drivers\msas2k3.sys (2.17.0.64 built by: WinDDK, 26.00 KB (26,624 bytes), 12/15/2006 2:25 PM)

[IDE]

Item Value
Name Standard Dual Channel PCI IDE Controller
Manufacturer (Standard IDE ATA/ATAPI controllers)
Status OK
PNP Device ID
PCI\VEN_8086&DEV_27DF&SUBSYS_03811014&REV_01\3&29D18217&0&F9
I/O Port 0x00000700-0x0000070F
Driver c:\windows\system32\drivers\pciide.sys (5.2.3790.1830 (srv03_sp1_rtm.050324-1447), 6.00 KB (6,144 bytes), 3/25/2005 7:00 AM)

Name Primary IDE Channel
Manufacturer (Standard IDE ATA/ATAPI controllers)
Status OK
PNP Device ID PCIIDE\IDECHANNEL\4&37BDDB0B&0&0
I/O Port 0x000001F0-0x000001F7
I/O Port 0x000003F6-0x000003F6
IRQ Channel IRQ 14
Driver c:\windows\system32\drivers\atapi.sys (5.2.3790.3959 (srv03_sp2_rtm.070216-1710), 146.50 KB (150,016 bytes), 3/25/2005 7:00 AM)

Name Secondary IDE Channel
Manufacturer (Standard IDE ATA/ATAPI controllers)
Status OK
PNP Device ID PCIIDE\IDECHANNEL\4&37BDDB0B&0&1
I/O Port 0x00000170-0x00000177
I/O Port 0x00000376-0x00000376
Driver c:\windows\system32\drivers\atapi.sys (5.2.3790.3959 (srv03_sp2_rtm.070216-1710), 146.50 KB (150,016 bytes), 3/25/2005 7:00 AM)

[Printing]

Name Driver Port Name Server Name

[Problem Devices]

Device PNP Device ID Error Code
HID SYS USB\VID_04B3&PID_4001&MI_02\6&B4B3E6A&0&0002 This device is disabled.

[USB]

Device PNP Device ID
Standard Universal PCI to USB Host Controller
PCI\VEN_8086&DEV_27C8&SUBSYS_03811014&REV_01\3&29D18217&0&E8
Standard Universal PCI to USB Host Controller
PCI\VEN_8086&DEV_27C9&SUBSYS_03811014&REV_01\3&29D18217&0&E9
Standard Universal PCI to USB Host Controller
PCI\VEN_8086&DEV_27CA&SUBSYS_03811014&REV_01\3&29D18217&0&EA
Standard Universal PCI to USB Host Controller
PCI\VEN_8086&DEV_27CB&SUBSYS_03811014&REV_01\3&29D18217&0&EB
Standard Enhanced PCI to USB Host Controller
PCI\VEN_8086&DEV_27CC&SUBSYS_03811014&REV_01\3&29D18217&0&EF
Standard Universal PCI to USB Host Controller
PCI\VEN_8086&DEV_27C8&SUBSYS_03811014&REV_01\3&136BC62F&0&E8
Standard Universal PCI to USB Host Controller
PCI\VEN_8086&DEV_27C9&SUBSYS_03811014&REV_01\3&136BC62F&0&E9
Standard Universal PCI to USB Host Controller
PCI\VEN_8086&DEV_27CA&SUBSYS_03811014&REV_01\3&136BC62F&0&EA
Standard Universal PCI to USB Host Controller
PCI\VEN_8086&DEV_27CB&SUBSYS_03811014&REV_01\3&136BC62F&0&EB
Standard Enhanced PCI to USB Host Controller
PCI\VEN_8086&DEV_27CC&SUBSYS_03811014&REV_01\3&136BC62F&0&EF

[Software Environment]

[System Drivers]

Name Description File Type Started Start ModeState
Status Error Control Accept Pause Accept Stop
abiosdsk Abiosdsk Not Available Kernel Driver No
Disabled Stopped OK Ignore No No
acpi Microsoft ACPI Driver c:\windows\system32\drivers\acpi.sys
Kernel Driver Yes Boot Running OK Normal No
Yes
acpiec ACPIEC c:\windows\system32\drivers\acpiec.sys Kernel Driver
No Disabled Stopped OK Normal No No
adpu160m adpu160m Not Available Kernel Driver No
Disabled Stopped OK Normal No No
adpu320 adpu320 Not Available Kernel Driver No
Disabled Stopped OK Normal No No
afd AFD c:\windows\system32\drivers\afd.sys Kernel Driver
Yes System Running OK Normal No Yes
aic78u2 aic78u2 Not Available Kernel Driver No
Disabled Stopped OK Normal No No
aic78xx aic78xx Not Available Kernel Driver No
Disabled Stopped OK Normal No No
aliide AliIde Not Available Kernel Driver No
Disabled Stopped OK Normal No No

amdide AmdIde Not Available Kernel Driver No
Disabled Stopped OK Normal No No
arc arc Not Available Kernel Driver No
Disabled Stopped OK Normal No No
asyncmac RAS Asynchronous Media Driver
c:\windows\system32\drivers\asyncmac.sys Kernel Driver No Manual
Stopped OK Normal No No
atapi Standard IDE/ESDI Hard Disk Controller
c:\windows\system32\drivers\atapi.sys Kernel Driver Yes Boot
Running OK Normal No Yes
atdisk Atdisk Not Available Kernel Driver No
Disabled Stopped OK Ignore No No
atmarpc ATM ARP Client Protocol c:\windows\system32\drivers\atmarpc.sys
Kernel Driver No Manual Stopped OK Normal No
No
audstub Audio Stub Driver c:\windows\system32\drivers\audstub.sys Kernel
Driver Yes Manual Running OK Normal No Yes
b06bdrv Broadcom NetXtreme II VBD c:\windows\system32\drivers\bxvbda.sys
Kernel Driver Yes Boot Running OK Normal No
Yes
beep Beep c:\windows\system32\drivers\beep.sys Kernel Driver
Yes System Running OK Normal No Yes
cdac15ba CdaC15BA c:\windows\system32\drivers\cdac15ba.sys Kernel
Driver Yes Auto Running OK Normal No Yes
cdad10ba CdaD10BA c:\windows\system32\drivers\cdad10ba.sys Kernel
Driver Yes Auto Running OK Normal No Yes
cdfs Cdfs c:\windows\system32\drivers\cdfs.sys File System Driver
Yes Disabled Running OK Normal No Yes
cdrom CD-ROM Driver c:\windows\system32\drivers\cdrom.sys Kernel
Driver Yes System Running OK Normal No Yes
changer Changer Not Available Kernel Driver No System
Stopped OK Ignore No No
clusdisk Cluster Disk Driver c:\windows\system32\drivers\clusdisk.sys Kernel
Driver No Disabled Stopped OK Normal No No
cmdide CmdIde Not Available Kernel Driver No
Disabled Stopped OK Normal No No
cpqcissm cpqcissm Not Available Kernel Driver No
Disabled Stopped OK Normal No No
crcdisk CRC Disk Filter Driver c:\windows\system32\drivers\crcdisk.sys
Kernel Driver Yes Boot Running OK Normal No
Yes
dfsdriver DfsDriver c:\windows\system32\drivers\dfs.sys File System Driver
Yes Boot Running OK Normal No Yes
disk Disk Driver c:\windows\system32\drivers\disk.sys Kernel
Driver Yes Boot Running OK Normal No Yes
dmboot dmboot c:\windows\system32\drivers\dmboot.sys Kernel Driver
No Disabled Stopped OK Normal No No
dmio Logical Disk Manager Driver c:\windows\system32\drivers\dmio.sys
Kernel Driver Yes Boot Running OK Normal No
Yes
dmload dmload c:\windows\system32\drivers\dmload.sys Kernel Driver
Yes Boot Running OK Normal No Yes
dpti2o dpti2o Not Available Kernel Driver No
Disabled Stopped OK Normal No No
e1express Intel(R) PRO/1000 PCI Express Network Connection Driver
c:\windows\system32\drivers\e1e5132e.sys Kernel Driver No Manual
Stopped OK Normal No No
elxstor elxstor Not Available Kernel Driver No
Disabled Stopped OK Normal No No
em em \??\c:\windows\system32\drivers\em.sys Kernel Driver
No Manual Stopped OK Normal No No
fastfat Fastfat c:\windows\system32\drivers\fastfat.sys File System Driver
No Disabled Stopped OK Normal No No
fdc Fdc c:\windows\system32\drivers\fdc.sys Kernel Driver
No System Stopped OK Ignore No No
fips Fips c:\windows\system32\drivers\fips.sys Kernel Driver
Yes System Running OK Normal No Yes
flpydisk Flpydisk c:\windows\system32\drivers\flpydisk.sys Kernel Driver
No System Stopped OK Ignore No No
fltmgr FltMgr c:\windows\system32\drivers\fltmgr.sys File System Driver
Yes Boot Running OK Normal No Yes
ftdisk Volume Manager Driver c:\windows\system32\drivers\ftdisk.sys
Kernel Driver Yes Boot Running OK Normal No
Yes
gpc Generic Packet Classifier c:\windows\system32\drivers\msgpc.sys
Kernel Driver Yes Manual Running OK Normal No
Yes
hidusb Microsoft HID Class Driver c:\windows\system32\drivers\hidusb.sys
Kernel Driver Yes Manual Running OK Ignore No
Yes
hpcisss hpcisss Not Available Kernel Driver No
Disabled Stopped OK Normal No No
http HTTP c:\windows\system32\drivers\http.sys Kernel Driver
No Manual Stopped OK Normal No No
i2omgmt i2omgmt Not Available Kernel Driver No System
Stopped OK Normal No No
i8042prt i8042prt c:\windows\system32\drivers\i8042prt.sys Kernel Driver
No System Stopped OK Ignore No No
iirsp iirsp Not Available Kernel Driver No
Disabled Stopped OK Normal No No
imapi CD-Burning Filter Driver c:\windows\system32\drivers\imapi.sys
Kernel Driver Yes System Running OK Normal No
Yes
intelide IntelIde Not Available Kernel Driver No
Disabled Stopped OK Normal No No
intelppm Intel Processor Driver c:\windows\system32\drivers\intelppm.sys Kernel
Driver Yes Manual Running OK Normal No Yes
ip6fw IPv6 Windows Firewall Driver c:\windows\system32\drivers\ip6fw.sys
Kernel Driver No Manual Stopped OK Normal No
No
ipfilterdriver IP Traffic Filter Driver
c:\windows\system32\drivers\ipfltdrv.sys Kernel Driver No Manual
Stopped OK Normal No No
ipinip IP in IP Tunnel Driverc:\windows\system32\drivers\ipinip.sys Kernel
Driver No Manual Stopped OK Normal No No
ipnat IP Network Address Translator c:\windows\system32\drivers\ipnat.sys
Kernel Driver No Manual Stopped OK Normal No
No
ipsec IPSEC driver c:\windows\system32\drivers\ipsec.sys Kernel
Driver Yes System Running OK Normal No Yes
isapnp PnP ISA/EISA Bus Driver c:\windows\system32\drivers\isapnp.sys
Kernel Driver Yes Boot Running OK Critical No
Yes
kbdclass Keyboard Class Driver
c:\windows\system32\drivers\kbdclass.sys Kernel Driver Yes System
Running OK Normal No Yes
kbdhid Keyboard HID Driver c:\windows\system32\drivers\kbdhid.sys Kernel
Driver Yes System Running OK Ignore No Yes
ksecdd KSecDD c:\windows\system32\drivers\ksecdd.sys Kernel Driver
Yes Boot Running OK Normal No Yes
ksthunk Kernel Streaming WOW64 Thunk Service
c:\windows\system32\drivers\ksthunk.sys Kernel Driver Yes Manual
Running OK Normal No Yes
l2nd Broadcom NetXtreme II BXND
c:\windows\system32\drivers\bxnd52a.sys Kernel Driver Yes Manual
Running OK Normal No Yes
lp6nds35 lp6nds35 Not Available Kernel Driver No
Disabled Stopped OK Normal No No
lsi_sas lsi_sas c:\windows\system32\drivers\lsi_sas.sys Kernel Driver
Yes Boot Running OK Normal No Yes
mnmdd mnmdd c:\windows\system32\drivers\mnmdd.sys Kernel Driver
Yes System Running OK Ignore No Yes
modem Modem c:\windows\system32\drivers\modem.sys Kernel Driver
No Manual Stopped OK Ignore No No
mouclass Mouse Class Driver c:\windows\system32\drivers\mouclass.sys Kernel
Driver Yes System Running OK Normal No Yes
mouhid Mouse HID Driver c:\windows\system32\drivers\mouhid.sys Kernel
Driver Yes Manual Running OK Ignore No Yes
mountmgr Mount Point Manager c:\windows\system32\drivers\mountmgr.sys Kernel
Driver Yes Boot Running OK Normal No Yes
mraid35x mraid35x Not Available Kernel Driver No
Disabled Stopped OK Normal No No
mrxdav WebDav Client Redirector c:\windows\system32\drivers\mrxdav.sys
File System Driver No Manual Stopped OK Normal No
No
mrxsmb MRXSMB c:\windows\system32\drivers\mrxsmb.sys File System Driver
Yes System Running OK Normal No Yes

msas2k3 msas2k3 c:\windows\system32\drivers\msas2k3.sys Kernel Driver
Yes Boot Running OK Normal No Yes
msfs Msfs c:\windows\system32\drivers\msfs.sys File System Driver
Yes System Running OK Normal No Yes
mssmbios Microsoft System Management BIOS Driver
c:\windows\system32\drivers\mssmbios.sys Kernel Driver Yes Manual
Running OK Normal No Yes
mup Mup c:\windows\system32\drivers\mup.sys File System Driver
Yes Boot Running OK Normal No Yes
ndis NDIS System Driver c:\windows\system32\drivers\ndis.sys Kernel
Driver Yes Boot Running OK Normal No Yes
ndistapi Remote Access NDIS TAPI Driver
c:\windows\system32\drivers\ndistapi.sys Kernel Driver Yes Manual
Running OK Normal No Yes
ndisuio NDIS Usermode I/O Protocol c:\windows\system32\drivers\ndisuio.sys
Kernel Driver No Manual Stopped OK Normal No
No
ndiswan Remote Access NDIS WAN Driver
c:\windows\system32\drivers\ndiswan.sys Kernel Driver Yes Manual
Running OK Normal No Yes
ndproxy NDIS Proxy c:\windows\system32\drivers\ndproxy.sys Kernel
Driver Yes Manual Running OK Normal No Yes
netbios NetBIOS Interface c:\windows\system32\drivers\netbios.sys File
System Driver Yes System Running OK Normal No
Yes
netbt NetBios over Tcpip c:\windows\system32\drivers\netbt.sys Kernel
Driver Yes System Running OK Normal No Yes
nfrd960 nfrd960 Not Available Kernel Driver No
Disabled Stopped OK Normal No No
npfs Npfs c:\windows\system32\drivers\npfs.sys File System Driver
Yes System Running OK Normal No Yes
ntfs Ntfs c:\windows\system32\drivers\ntfs.sys File System Driver
Yes Disabled Running OK Normal No Yes
null Null c:\windows\system32\drivers\null.sys Kernel Driver
Yes System Running OK Normal No Yes
parport Parport c:\windows\system32\drivers\parport.sys Kernel Driver
No Manual Stopped OK Ignore No No
partmgr Partition Manager c:\windows\system32\drivers\partmgr.sys Kernel
Driver Yes Boot Running OK Normal No Yes
pci PCI Bus Driver c:\windows\system32\drivers\pci.sys Kernel
Driver Yes Boot Running OK Critical No Yes
pciide PCIIde c:\windows\system32\drivers\pciide.sys Kernel Driver
Yes Boot Running OK Normal No Yes
pcmcia Pcmcia c:\windows\system32\drivers\pcmcia.sys Kernel Driver
No Disabled Stopped OK Normal No No
pdcomp PDCOMP Not Available Kernel Driver No Manual
Stopped OK Ignore No No
pdframe PDFRAME Not Available Kernel Driver No
Manual Stopped OK Ignore No No
pdreli PDRELI Not Available Kernel Driver No Manual
Stopped OK Ignore No No
pdrframe PDRFRAME Not Available Kernel Driver No
Manual Stopped OK Ignore No No
pnpmem Microsoft Memory Module Driver
c:\windows\system32\drivers\pnpmem.sys Kernel Driver Yes Manual
Running OK Normal No Yes
pptpminiport WAN Miniport (PPTP)
c:\windows\system32\drivers\raspptp.sys Kernel Driver Yes Manual
Running OK Normal No Yes
ptilink Direct Parallel Link Driver c:\windows\system32\drivers\ptilink.sys
Kernel Driver Yes Manual Running OK Normal No
Yes
ql2300 ql2300 Not Available Kernel Driver No
Disabled Stopped OK Normal No No
rasacd Remote Access Auto Connection Driver
c:\windows\system32\drivers\rasacd.sys Kernel Driver Yes System
Running OK Normal No Yes
rasl2tp WAN Miniport (L2TP) c:\windows\system32\drivers\rasl2tp.sys
Kernel Driver Yes Manual Running OK Normal No
Yes
raspppoe Remote Access PPPOE Driver
c:\windows\system32\drivers\raspppoe.sys Kernel Driver Yes Manual
Running OK Normal No Yes
raspti Direct Parallel c:\windows\system32\drivers\raspti.sys Kernel
Driver Yes Manual Running OK Normal No Yes
rdbss Rdbss c:\windows\system32\drivers\rdbss.sys File System Driver
Yes System Running OK Normal No Yes
rdpcdd RDPCDD c:\windows\system32\drivers\rdpcdd.sys Kernel Driver
Yes System Running OK Ignore No Yes
rdpdr Terminal Server Device Redirector Driver
c:\windows\system32\drivers\rdpdr.sys Kernel Driver Yes Manual
Running OK Normal No Yes
rdpwd RDPWD c:\windows\system32\drivers\rdpwd.sys Kernel Driver
Yes Manual Running OK Ignore No Yes
redbook Digital CD Audio Playback Filter Driver
c:\windows\system32\drivers\redbook.sys Kernel Driver Yes System
Running OK Normal No Yes
secdrv Security Driver c:\windows\system32\drivers\secdrv.sys Kernel
Driver Yes Auto Running OK Normal No Yes
serenum Serenum Filter Driver c:\windows\system32\drivers\serenum.sys Kernel
Driver Yes Manual Running OK Normal No Yes
serial Serial port driver c:\windows\system32\drivers\serial.sys Kernel
Driver Yes System Running OK Ignore No Yes
sfloppy High-Capacity Floppy Disk Drive
c:\windows\system32\drivers\sfloppy.sys Kernel Driver No Manual
Stopped OK Normal No No
simbad Simbad Not Available Kernel Driver No
Disabled Stopped OK Normal No No
srv Srv c:\windows\system32\drivers\srv.sys File System Driver
Yes Manual Running OK Normal No Yes
swenum Software Bus Driver c:\windows\system32\drivers\swenum.sys Kernel
Driver Yes Manual Running OK Normal No Yes
symc8xx symc8xx Not Available Kernel Driver No
Disabled Stopped OK Normal No No
symmpi symmpi Not Available Kernel Driver No
Disabled Stopped OK Normal No No
sym_hi sym_hi Not Available Kernel Driver No
Disabled Stopped OK Normal No No
sym_u3 sym_u3 Not Available Kernel Driver No
Disabled Stopped OK Normal No No
tcpip TCP/IP Protocol Driver c:\windows\system32\drivers\tcpip.sys
Kernel Driver Yes System Running OK Normal No
Yes
tdpipe TDPIPE c:\windows\system32\drivers\tdpipe.sys Kernel Driver
No Manual Stopped OK Ignore No No
tdtcp TDTCP c:\windows\system32\drivers\tdtcp.sys Kernel Driver
Yes Manual Running OK Ignore No Yes
termdd Terminal Device Driver c:\windows\system32\drivers\termdd.sys
Kernel Driver Yes System Running OK Normal No
Yes
toside TosIde Not Available Kernel Driver No
Disabled Stopped OK Normal No No
udfs Udfs c:\windows\system32\drivers\udfs.sys File System Driver
No Disabled Stopped OK Normal No No
ultra ultra Not Available Kernel Driver No
Disabled Stopped OK Normal No No
update Microcode Update Driver c:\windows\system32\drivers\update.sys
Kernel Driver Yes Manual Running OK Normal No
Yes
usbccgp Microsoft USB Generic Parent Driver
c:\windows\system32\drivers\usbccgp.sys Kernel Driver Yes Manual
Running OK Normal No Yes
usbehci Microsoft USB 2.0 Enhanced Host Controller Miniport Driver
c:\windows\system32\drivers\usbehci.sys Kernel Driver Yes Manual
Running OK Normal No Yes
usbhub USB2 Enabled Hub c:\windows\system32\drivers\usbhub.sys Kernel
Driver Yes Manual Running OK Normal No Yes
usbstor USB Mass Storage Driver c:\windows\system32\drivers\usbstor.sys
Kernel Driver No Manual Stopped OK Normal No
No
usbuhci Microsoft USB Universal Host Controller Miniport Driver
c:\windows\system32\drivers\usbuhci.sys Kernel Driver Yes Manual
Running OK Normal No Yes
vga vga c:\windows\system32\drivers\vgapnp.sys Kernel Driver
Yes Manual Running OK Ignore No Yes

vgasave VGA Display Controller. c:\windows\system32\drivers\vga.sys
Kernel Driver No System Stopped OK Ignore No
No
viaide ViaIde Not Available Kernel Driver No
Disabled Stopped OK Normal No No
volsnap Storage volumes c:\windows\system32\drivers\volsnap.sys Kernel
Driver Yes Boot Running OK Normal No Yes
wanarp Remote Access IP ARP Driver c:\windows\system32\drivers\wanarp.sys
Kernel Driver Yes Manual Running OK Normal No
Yes
wdf01000 Wdf01000 c:\windows\system32\drivers\wdf01000.sys Kernel Driver
Yes Manual Running OK Normal No Yes
wdica WDICA Not Available Kernel Driver No Manual
Stopped OK Ignore No No
wlbs Network Load Balancing c:\windows\system32\drivers\wlbs.sys
Kernel Driver No Manual Stopped OK Normal No
No

[Signed Drivers]

Device Name Signed Device Class Driver Version Driver
Date Manufacturer INF Name Driver Name Device ID
Microsoft System Management BIOS DriverYes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
ROOT\SYSTEM\0002
Microcode Update Device Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
ROOT\SYSTEM\0001
Plug and Play Software Device Enumerator Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
ROOT\SYSTEM\0000
Terminal Server Mouse Driver Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
ROOT\RDP_MOU\0000
Terminal Server Keyboard Driver Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
ROOT\RDP_KBD\0000
Terminal Server Device Redirector Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
ROOT\RDPDR\0000
Direct Parallel Yes NET 5.2.3790.1830 10/1/2002
Microsoft netrasa.inf Not Available ROOT\MS_PTIMINIPORT\0000
WAN Miniport (PPTP) Yes NET 5.2.3790.1830
10/1/2002 Microsoft netrasa.inf Not Available
ROOT\MS_PPTPMINIPORT\0000
WAN Miniport (PPPOE) Yes NET 5.2.3790.1830
10/1/2002 Microsoft netrasa.inf Not Available
ROOT\MS_PPPOEMINIPORT\0000
WAN Miniport (IP) Yes NET 5.2.3790.1830 10/1/2002
Microsoft netrasa.inf Not Available ROOT\MS_NDISWANIP\0000
WAN Miniport (L2TP) Yes NET 5.2.3790.1830
10/1/2002 Microsoft netrasa.inf Not Available
ROOT\MS_L2TPMINIPORT\0000
Video Codecs Yes MEDIA 5.2.3790.1830 10/1/2002
(Standard system devices) wave.inf Not Available
ROOT\MEDIA\MS_MMVID
Legacy Video Capture Devices Yes MEDIA 5.2.3790.1830
10/1/2002 (Standard system devices) wave.inf Not Available
ROOT\MEDIA\MS_MMVCD
Media Control Devices Yes MEDIA 5.2.3790.1830
10/1/2002 (Standard system devices) wave.inf Not Available
ROOT\MEDIA\MS_MMMCI
Legacy Audio Drivers Yes MEDIA 5.2.3790.1830 10/1/2002
(Standard system devices) wave.inf Not Available
ROOT\MEDIA\MS_MMDRV
Audio Codecs Yes MEDIA 5.2.3790.1830 10/1/2002
(Standard system devices) wave.inf Not Available
ROOT\MEDIA\MS_MMACM
Remote Access IP ARP Driver Not Available LEGACYDRIVER Not
Available Not Available Not Available Not Available Not
Available ROOT\LEGACY_WANARP\0000
volsnap Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_VOLSNAP\0000
TDTCP Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_TDTCP\0000
TCP/IP Protocol Driver Not Available LEGACYDRIVER Not
Available Not Available Not Available Not Available Not
Available ROOT\LEGACY_TCPIP\0000
Security Driver Not Available LEGACYDRIVER Not Available
Not Available Not Available Not Available Not Available
ROOT\LEGACY_SECDRV\0000
RDPWD Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_RDPWD\0000
RDPCDD Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_RDPCDD\0000
Remote Access Auto Connection Driver Not Available LEGACYDRIVER
Not Available Not Available Not Available Not Available
Not Available ROOT\LEGACY_RASACD\0000
Partition Manager Not Available LEGACYDRIVER Not Available
Not Available Not Available Not Available Not Available
ROOT\LEGACY_PARTMGR\0000
Null Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_NULL\0000
NetBios over Tcpip Not Available LEGACYDRIVER Not Available
Not Available Not Available Not Available Not Available
ROOT\LEGACY_NETBT\0000
NDProxy Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_NDPROXY\0000
NDIS Usermode I/O Protocol Not Available LEGACYDRIVER Not
Available Not Available Not Available Not Available Not
Available ROOT\LEGACY_NDISUIO\0000
Remote Access NDIS TAPI Driver Not Available LEGACYDRIVER
Not Available Not Available Not Available Not Available
Not Available ROOT\LEGACY_NDISTAPI\0000
NDIS System Driver Not Available LEGACYDRIVER Not Available
Not Available Not Available Not Available Not Available
ROOT\LEGACY_NDIS\0000
mountmgr Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_MOUNTMGR\0000
mnmdd Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_MNMDD\0000
ksecdd Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_KSECDD\0000
IPSEC driver Not Available LEGACYDRIVER Not Available
Not Available Not Available Not Available Not Available
ROOT\LEGACY_IPSEC\0000
IP Network Address Translator Not Available LEGACYDRIVER Not
Available Not Available Not Available Not Available Not
Available ROOT\LEGACY_IPNAT\0000
Generic Packet Classifier Not Available LEGACYDRIVER Not
Available Not Available Not Available Not Available Not
Available ROOT\LEGACY_GPC\0000
Fips Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_FIPS\0000
dmload Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_DMLOAD\0000
dmboot Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_DMBOOT\0000
CRC Disk Filter Driver Not Available LEGACYDRIVER Not
Available Not Available Not Available Not Available Not
Available ROOT\LEGACY_CRCDISK\0000
CdaD10BA Not Available LEGACYDRIVER Not Available
Not Available Not Available Not Available Not Available
ROOT\LEGACY_CDAD10BA\0000

CdaC15BA Not Available LEGACYDRIVER Not Available
Not Available Not Available Not Available Not Available
ROOT\LEGACY_CDAC15BA\0000
Beep Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_BEEP\0000
AFD Not Available LEGACYDRIVER Not Available Not
Available Not Available Not Available Not Available
ROOT\LEGACY_AFD\0000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C6FOFFSET3FF560E
800LENGTH500146800
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C61OFFSET3FF560E
800LENGTH500146800
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C6FOFFSET3AF54C0
200LENGTH500146800
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C61OFFSET3AF54C0
200LENGTH500146800
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CBBOFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CBBOFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CBBOFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CBBOFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CBBOFFSET7E0000
LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CCDOFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CCDOFFSETE7502F
800LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CCDOFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CCDOFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CCDOFFSET7E0000
LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C73OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C73OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C73OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C73OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C73OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C85OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C85OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C85OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C85OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C85OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C4BOFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C4BOFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C4BOFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C4BOFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C4BOFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C5DOFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C5DOFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C5DOFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C5DOFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C5DOFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available

STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C03OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C03OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C03OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C03OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C03OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C15OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C15OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C15OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C15OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C15OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DDBOFFSETE7502F
800LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DDBOFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DDBOFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DDBOFFSET7E0000
LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DEDOFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DEDOFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DEDOFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DEDOFFSET7E0000
LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE55912642OFFSETE7502F80
0LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE55912642OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE55912642OFFSET5AAB21C
00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE55912642OFFSET7E0000LE
NGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DA5OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DA5OFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DA5OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DA5OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DB7OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DB7OFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DB7OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DB7OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DC9OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DC9OFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DC9OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DC9OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DFFOFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DFFOFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DFFOFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available

STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DFFOFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DF1OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DF1OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DF1OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1DF1OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C27OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C27OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C27OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C27OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C27OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C39OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C39OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C39OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C39OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C39OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURED392CE83OFFSET35E972E
C00LENGTHC7893000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURED392CE83OFFSET1CE90A6
E00LENGTH1900680000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURED392CE83OFFSET1900E67
E00LENGTH3E8237200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURED392CE83OFFSET7E0000L
ENGTH1900680000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C6FOFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C6FOFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C6FOFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C6FOFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C6FOFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C61OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C61OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C61OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C61OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C61OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C97OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C97OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C97OFFSET8CACED
C00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C97OFFSET5AAB21
C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1C97OFFSET7E0000L
ENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CA9OFFSET12F5225
400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CA9OFFSETE7502F8
00LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CA9OFFSET8CACE
DC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available

STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CA9OFFSET5AAB21C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CA9OFFSET7E0000LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CDFOFFSET12F5225400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CDFOFFSETE7502F800LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CDFOFFSET8CACEDC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CDFOFFSET5AAB21C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CDFOFFSET7E0000LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CD1OFFSET12F5225400LENGTH2800293000
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CD1OFFSETE7502F800LENGTH4801EDE00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CD1OFFSET8CACEDC00LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CD1OFFSET5AAB21C00LENGTH3201C4200
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATURE46CC1CD1OFFSET7E0000LENGTH5AA339E00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATUREE21AE21BOFFSET7E00LENGTH11171F3A00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATUREE21AE21AOFFSET7E00LENGTH11171F3A00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATUREE101E101OFFSET9003EB800LENGTH7C30B2A00
Generic volume Yes VOLUME 5.2.3790.1830 10/1/2002
Microsoft volume.inf Not Available
STORAGE\VOLUME\1&30A96598&0&SIGNATUREE101E101OFFSET7E00LENGTH9003E3A00
Volume Manager Yes SYSTEM 5.2.3790.1830 10/1/2002
(Standard system devices) machine.inf Not Available
ROOT\FTDISK\0000
Logical Disk ManagerYes SYSTEM 5.2.3790.1830 10/1/2002
(Standard system devices) machine.inf Not Available
ROOT\DMIO\0000
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&28283CC0&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&28283CC0&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&28283CC0&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&28283CC0&0&005500
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&28283CC0&0&004800
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&10B57C39&0&0000
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&262576D4&0&00
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\17
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&338DFEE0&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&338DFEE0&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&338DFEE0&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&338DFEE0&0&002100
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&338DFEE0&0&001400
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&36CC30BC&0&0000
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&38DF2779&0&00
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\16
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&1CBA9F17&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&1CBA9F17&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&1CBA9F17&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0151\5&1CBA9F17&0&002100
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&1CBA9F17&0&001400

IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&A41217E&0&0000
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&FFE0E17&0&00
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\15
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&22B7BEBC&0&00
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\14
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&59F3E4
F&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&59F3E4F
&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&59F3E4F
&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&59F3E4F
&0&002100
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&59F3E4F
&0&001400
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&2111E872&0&000
0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&35716F61&0&00
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\13
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&25495E
9F&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&25495E9F
&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&25495E9F
&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&25495E9
F&0&002E00
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&25495E9
F&0&002100
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&13C9039C&0&000
0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&50481E5&0&00
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\12
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&AF336B
4&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&AF336B4
&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&AF336B4
&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&AF336B
4&0&002E00
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&AF336B
4&0&001400
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&3ED7AD6&0&000
0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&2DE59B47&0&00
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\11
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&1B2BEAA2&0&00
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\10
PCI standard ISA bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_8086&DEV_27B8&SUBSYS_00000000&REV_01\3&136BC62F&0&F8
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB20\4&91A3D59&0
Standard Enhanced PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27CC&SUBSYS_03811014&REV_01\3&136BC62F&0&EF
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB\4&21F73DE4&0
Standard Universal PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27CB&SUBSYS_03811014&REV_01\3&136BC62F&0&EB
HID-compliant device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
HID\VID_04B3&PID_4001&MI_02\7&13F1D0FA&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_04B3&PID_4001&MI_02\6&15A8C041&0&0002
HID-compliant mouse Yes MOUSE 5.2.3790.1830 10/1/2002
Microsoft msmouse.inf Not Available
HID\VID_04B3&PID_4001&MI_01\7&37C90EBC&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_04B3&PID_4001&MI_01\6&15A8C041&0&0001
HID Keyboard Device Yes KEYBOARD 5.2.3790.1830
10/1/2002 (Standard keyboards) keyboard.inf Not Available
HID\VID_04B3&PID_4001&MI_00\7&20058277&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_04B3&PID_4001&MI_00\6&15A8C041&0&0000
USB Composite Device Yes USB 5.2.3790.1830
10/1/2002 (Standard USB Host Controller) usb.inf Not Available
USB\VID_04B3&PID_4001\00145ECF5CF9
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB\4&37668657&0

Standard Universal PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27CA&SUBSYS_03811014&REV_01\3&136BC62F&0&EA
HID-compliant device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
HID\VID_0624&PID_0200&MI_01&COL03\7&12889A6&0&0002
HID-compliant consumer control device Yes HIDCLASS
5.2.3790.1830 10/1/2002 Microsoft hidserv.inf Not Available
HID\VID_0624&PID_0200&MI_01&COL02\7&12889A6&0&0001
HID-compliant mouse Yes MOUSE 5.2.3790.1830 10/1/2002
Microsoft msmouse.inf Not Available
HID\VID_0624&PID_0200&MI_01&COL01\7&12889A6&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_0624&PID_0200&MI_01\6&20E99C89&0&0001
HID Keyboard Device Yes KEYBOARD 5.2.3790.1830
10/1/2002 (Standard keyboards) keyboard.inf Not Available
HID\VID_0624&PID_0200&MI_00\7&24FFC768&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_0624&PID_0200&MI_00\6&20E99C89&0&0000
USB Composite Device Yes USB 5.2.3790.1830
10/1/2002 (Standard USB Host Controller) usb.inf Not Available
USB\VID_0624&PID_0200\5&4C204AB&0&1
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB\4&4AE1BD2&0
Standard Universal PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27C9&SUBSYS_03811014&REV_01\3&136BC62F&0&E9
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB\4&10C12CA1&0
Standard Universal PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27C8&SUBSYS_03811014&REV_01\3&136BC62F&0&E8
Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client) No NET
3.7.4.0 6/11/2007 Broadcom Corporation oem7.inf Not Available
B06BDRV\L2ND&PCI_163914E4&SUBSYS_037C1014&REV_01\5&20CCE83A&
0&20053100
Broadcom BCM5709C NetXtreme II GigE No SYSTEM 3.7.5.0
6/25/2007 Broadcom Corporation oem6.inf Not Available
PCI\VEN_14E4&DEV_1639&SUBSYS_037C1014&REV_01\4&FABD7BC&0&01
E0
Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client) No NET
3.7.4.0 6/11/2007 Broadcom Corporation oem7.inf Not Available
B06BDRV\L2ND&PCI_163914E4&SUBSYS_037C1014&REV_01\5&833C090&0
&20053100
Broadcom BCM5709C NetXtreme II GigE No SYSTEM 3.7.5.0
6/25/2007 Broadcom Corporation oem6.inf Not Available
PCI\VEN_14E4&DEV_1639&SUBSYS_037C1014&REV_01\4&FABD7BC&0&00
E0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_8086&DEV_27D0&SUBSYS_00000000&REV_01\3&136BC62F&0&E0
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\18
Memory Module Yes MEMORY5.2.3790.1830 10/1/2002
Microsoft memory.inf Not Available ACPI\PNP0C80\2
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\31
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\30
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\29
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\28
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\27
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\26
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\25
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\24
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\23
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\22
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\21
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\20
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\19
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\18
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\17
Intel Processor Yes PROCESSOR 5.2.3790.1830
10/1/2002 Intel cpu.inf Not Available
ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\16
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&2A7839
C9&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&2A7839C9
&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&2A7839C9
&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&2A7839
C9&0&002100
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&2A7839
C9&0&001400
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&2657B4F5&0&000
0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&2FF92B1D&0&00
PCI bus Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\7
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&136B68
65&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&136B6865
&0&010100

Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&136B6865
&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&136B686
5&0&002100
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&136B686
5&0&001400
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&167C2C2F&0&000
0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&71811BB&0&00
PCI bus Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\6
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&297181
F&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&297181F&
0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&297181F&
0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&297181F
&0&002100
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&297181F
&0&001400
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&1DD220B0&0&00
00
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&19D1C260&0&00
PCI bus Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\5
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&35C517
76&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&35C51776
&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&35C5177
6&0&002100
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&394070D2&0&000
0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&2C8B7305&0&00
PCI bus Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\4
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&37E06A
26&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&37E06A26
&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&37E06A26
&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&37E06A2
6&0&002100
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&37E06A2
6&0&001400
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&98F6FDA&0&0000
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&3AA59A3&0&00
PCI bus Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\3
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&12CE92
A&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&12CE92A
&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&12CE92A
&0&010000
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&15BF98E9&0&000
0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&36CB97A3&0&00
PCI bus Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\2
RAID Virtual Device Yes SYSTEM 5.2.3790.1830 10/1/2002
American Megatrends, Inc. scsidev.inf Not Available
SCSI\OTHER&VEN___RAID&PROD__DUMMYDEVICE&REV_0001\5&348F13
FF&0&014000
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&348F13FF
&0&010100
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002
(Standard disk drives) disk.inf Not Available
SCSI\DISK&VEN_IBM&PROD_MEGARAID_8880EM2&REV_1.12\5&348F13FF
&0&010000
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&348F13F
F&0&002100
IBM-ESXS_EXP3000 Enclosure Device No SYSTEM 2.17.0.64
8/13/2007 LSI Corp. oem9.inf Not Available
SCSI\ENCLOSURE&VEN_IBM-ESXS&PROD_EXP3000&REV_0166\5&348F13F
F&0&001400
IBM SystemX MegaRAID SAS 8880EM2 RAID Controller Yes
SCSIADAPTER 2.17.0.64 8/13/2007 LSI Corp., oem8.inf Not Available
PCI\VEN_1000&DEV_0060&SUBSYS_03791014&REV_04\4&1DD35BBD&0&00
00
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.1830
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&2411E6FE&0&00

PCI bus Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0A08\1
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002 (Standard disk drives) disk.inf Not Available SCSI\DISK&VEN_IBM-ESXS&PROD_ST973401SS&REV_B51C\5&22AD4189&0&000B00
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002 (Standard disk drives) disk.inf Not Available SCSI\DISK&VEN_IBM-ESXS&PROD_MAY2073RC&REV_T106\5&22AD4189&0&000A00
Disk drive Yes DISKDRIVE 5.2.3790.1830 10/1/2002 (Standard disk drives) disk.inf Not Available SCSI\DISK&VEN_LSILOGIC&PROD_LOGICAL_VOLUME&REV_3000\5&22AD4189&0&000800
IBM SAS SES DeviceYes SYSTEM 1.0.1.0 12/8/2006 IBM Corporation oem0.inf Not Available SCSI\ENCLOSURE&VEN_IBM&PROD_SAS_SES-2_DEVICE&REV_01.0\5&22AD4189&0&000000
LSI Adapter, SAS RAID-on-Chip, 8-port with 1078 -StorPort No SCSIADAPTER 1.25.10.7 9/20/2007 LSI Corporation oem10.inf Not Available PCI\VEN_1000&DEV_0062&SUBSYS_10001000&REV_03\4&2BCCA3F5&0&0000
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available PCI\VEN_1014&DEV_0308&SUBSYS_00000000&REV_01\3&11583659&0&00
PCI bus Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0A08\0
Memory Module Yes MEMORY5.2.3790.1830 10/1/2002 Microsoft memory.inf Not Available ACPI\PNP0C80\0
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\15
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\14
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\13
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\12
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\11
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15\10
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_9
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_8
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_7
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_6
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_5
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_4
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_3
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_2
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_1
Intel Processor Yes PROCESSOR 5.2.3790.1830 10/1/2002 Intel cpu.inf Not Available ACPI\GENUINEINTEL_-_EM64T_FAMILY_6_MODEL_15_0
ACPI Fixed Feature Button Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\FIXEDBUTTON\2&DABA3FF&0
Motherboard resources Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0C31\1
Motherboard resources Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0C02\D
System board Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0C01\1
Motherboard resources Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\IBM37D4\0
Secondary IDE Channel Yes HDC 5.2.3790.1830 10/1/2002 (Standard IDE ATA/ATAPI controllers) mshdc.inf Not Available PCIIDE\IDECHANNEL\4&37BDDB0B&0&1
CD-ROM Drive Yes CDROM 5.2.3790.1830 10/1/2002 (Standard CD-ROM drives) cdrom.inf Not Available IDE\CDROMHL-DT-ST_RW/DVD_GCC-T10N_____1.00____\5&E769355&0&0.0.0
Primary IDE Channel Yes HDC 5.2.3790.1830 10/1/2002 (Standard IDE ATA/ATAPI controllers) mshdc.inf Not Available PCIIDE\IDECHANNEL\4&37BDDB0B&0&0
Standard Dual Channel PCI IDE Controller Yes HDC 5.2.3790.1830 10/1/2002 (Standard IDE ATA/ATAPI controllers) mshdc.inf Not Available PCI\VEN_8086&DEV_27DF&SUBSYS_03811014&REV_01\3&29D18217&0&F9
Motherboard resources Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\IPI0001\0
Motherboard resources Yes SYSTEM 5.2.3790.1830 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0C02\3
Numeric data processor Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0C04\4&3695E41E&0
System speaker Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0800\4&3695E41E&0
System CMOS/real time clock Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0B00\4&3695E41E&0
System timer Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0100\4&3695E41E&0
Direct memory access controller Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0200\4&3695E41E&0
Advanced programmable interrupt controller Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available ACPI\PNP0003\4&3695E41E&0
Communications Port Yes PORTS 5.2.3790.1830 10/1/2002 (Standard port types) msports.inf Not Available ACPI\PNP0501\1
PCI standard ISA bridge Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system devices) machine.inf Not Available PCI\VEN_8086&DEV_27B8&SUBSYS_00000000&REV_01\3&29D18217&0&F8
Plug and Play Monitor Yes MONITOR 5.2.3790.1830 10/1/2002 (Standard monitor types) monitor.inf Not Available DISPLAY\AVO0402\5&9D68E52&0&12345678&01&00
Standard VGA Graphics Adapter Yes DISPLAY 5.2.3790.1830 10/1/2002 (Standard display types) display.inf Not Available PCI\VEN_1002&DEV_515E&SUBSYS_03191014&REV_02\4&18E510B5&0&00F0
Intel(R) 82801 PCI Bridge - 244E Yes SYSTEM 5.2.3790.3959 10/1/2002 Intel machine.inf Not Available PCI\VEN_8086&DEV_244E&SUBSYS_00000000&REV_E1\3&29D18217&0&F0

USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB20\4&CEBC86B&0
Standard Enhanced PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27CC&SUBSYS_03811014&REV_01\3&29D18217&0&EF
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB\4&25C8C8F6&0
Standard Universal PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27CB&SUBSYS_03811014&REV_01\3&29D18217&0&EB
HID SYS Not Available HIDCLASS Not Available Not
Available (Standard system devices) Not Available Not Available
USB\VID_04B3&PID_4001&MI_02\6&B4B3E6A&0&0002
HID-compliant mouse Yes MOUSE 5.2.3790.1830 10/1/2002
Microsoft msmouse.inf Not Available
HID\VID_04B3&PID_4001&MI_01\7&11AE42E5&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_04B3&PID_4001&MI_01\6&B4B3E6A&0&0001
HID Keyboard Device Yes KEYBOARD 5.2.3790.1830
10/1/2002 (Standard keyboards) keyboard.inf Not Available
HID\VID_04B3&PID_4001&MI_00\7&3A18F0AE&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_04B3&PID_4001&MI_00\6&B4B3E6A&0&0000
USB Composite Device Yes USB 5.2.3790.1830
10/1/2002 (Standard USB Host Controller) usb.inf Not Available
USB\VID_04B3&PID_4001\00145ECF5CC8
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB\4&11F79082&0
Standard Universal PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27CA&SUBSYS_03811014&REV_01\3&29D18217&0&EA
HID-compliant device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
HID\VID_0624&PID_0200&MI_01&COL03\7&5A1E937&0&0002
HID-compliant consumer control device Yes HIDCLASS
5.2.3790.1830 10/1/2002 Microsoft hidserv.inf Not Available
HID\VID_0624&PID_0200&MI_01&COL02\7&5A1E937&0&0001
HID-compliant mouse Yes MOUSE 5.2.3790.1830 10/1/2002
Microsoft msmouse.inf Not Available
HID\VID_0624&PID_0200&MI_01&COL01\7&5A1E937&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_0624&PID_0200&MI_01\6&3675DEE2&0&0001
HID Keyboard Device Yes KEYBOARD 5.2.3790.1830
10/1/2002 (Standard keyboards) keyboard.inf Not Available
HID\VID_0624&PID_0200&MI_00\7&1D65757C&0&0000
USB Human Interface Device Yes HIDCLASS 5.2.3790.1830
10/1/2002 (Standard system devices) input.inf Not Available
USB\VID_0624&PID_0200&MI_00\6&3675DEE2&0&0000
USB Composite Device Yes USB 5.2.3790.1830
10/1/2002 (Standard USB Host Controller) usb.inf Not Available
USB\VID_0624&PID_0200\5&2A12A76F&0&1
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB\4&3ABE3947&0
Standard Universal PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27C9&SUBSYS_03811014&REV_01\3&29D18217&0&E9
USB Root Hub Yes USB 5.2.3790.1830 10/1/2002
(Standard USB Host Controller) usbport.inf Not Available
USB\ROOT_HUB\4&389CEA38&0
Standard Universal PCI to USB Host Controller Yes USB
5.2.3790.1830 10/1/2002 (Standard USB Host Controller) usbport.inf Not
Available
PCI\VEN_8086&DEV_27C8&SUBSYS_03811014&REV_01\3&29D18217&0&E8
Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client) No NET
3.7.4.0 6/11/2007 Broadcom Corporation oem7.inf Not Available
B06BDRV\L2ND&PCI_163914E4&SUBSYS_037C1014&REV_01\5&38892308&0
&20050200
Broadcom BCM5709C NetXtreme II GigE No SYSTEM 3.7.5.0
6/25/2007 Broadcom Corporation oem6.inf Not Available
PCI\VEN_14E4&DEV_1639&SUBSYS_037C1014&REV_01\4&39B23F1D&0&01
E0
Broadcom BCM5709C NetXtreme II GigE (NDIS VBD Client) No NET
3.7.4.0 6/11/2007 Broadcom Corporation oem7.inf Not Available
B06BDRV\L2ND&PCI_163914E4&SUBSYS_037C1014&REV_01\5&158780AB&
0&20050200
Broadcom BCM5709C NetXtreme II GigE No SYSTEM 3.7.5.0
6/25/2007 Broadcom Corporation oem6.inf Not Available
PCI\VEN_14E4&DEV_1639&SUBSYS_037C1014&REV_01\4&39B23F1D&0&00
E0
PCI standard PCI-to-PCI bridge Yes SYSTEM 5.2.3790.3959
10/1/2002 (Standard system devices) machine.inf Not Available
PCI\VEN_8086&DEV_27D0&SUBSYS_00000000&REV_01\3&29D18217&0&E0
PCI bus Yes SYSTEM 5.2.3790.3959 10/1/2002 (Standard system
devices) machine.inf Not Available ACPI\PNP0A08\E
Microsoft ACPI-Compliant System Yes SYSTEM 5.2.3790.1830
10/1/2002 Microsoft acpi.inf Not Available ACPI_HAL\PNP0C08\0
ACPI Multiprocessor x64-based PC Yes COMPUTER
5.2.3790.1830 10/1/2002 (Standard computers) hal.inf Not Available
ROOT\ACPI_HAL\0000
Not Available Not Available Not Available Not Available
Not Available Not Available Not Available Not Available
HTREE\ROOT\0

[Environment Variables]

Variable Value User Name
ClusterLog C:\WINDOWS\Cluster\cluster.log <SYSTEM>
ComSpec %SystemRoot%\system32\cmd.exe <SYSTEM>
FP_NO_HOST_CHECK NO <SYSTEM>
lib C:\Program Files\SQLXML 4.0\bin\ <SYSTEM>
NUMBER_OF_PROCESSORS 32 <SYSTEM>
OS Windows_NT <SYSTEM>
Path
%SystemRoot%\system32;%SystemRoot%;%SystemRoot%\System32\Wbem;c:\batfil
es;c:\tools;c:\tools\util;C:\Program Files\Intel\DMIX;C:\Program Files
(x86)\Microsoft SQL Server\80\Tools\Binn\;C:\Program Files\Microsoft SQL
Server\90\Tools\binn\;C:\Program Files (x86)\Microsoft SQL
Server\90\Tools\binn\;C:\Program Files (x86)\Microsoft SQL
Server\90\DTS\Binn\;C:\Program Files (x86)\Microsoft SQL
Server\90\Tools\Binn\VSShell\Common7\IDE\;C:\Program Files (x86)\Microsoft
Visual Studio 8\Common7\IDE\PrivateAssemblies\;c:\Program Files\Microsoft SQL
Server\90\DTS\Binn\;C:\Program Files\Microsoft SQL
Server\MSSQL.1\MSSQL\Binn <SYSTEM>
PATHEXT .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH
<SYSTEM>
PROCESSOR_ARCHITECTURE AMD64 <SYSTEM>
PROCESSOR_IDENTIFIER EM64T Family 6 Model 15 Stepping 11,
GenuineIntel <SYSTEM>
PROCESSOR_LEVEL 6 <SYSTEM>
PROCESSOR_REVISION 0f0b <SYSTEM>
TEMP %SystemRoot%\TEMP <SYSTEM>
TMP %SystemRoot%\TEMP <SYSTEM>
windir %SystemRoot% <SYSTEM>
TPCH_SCALE_FACTOR_SIMPLE 300 <SYSTEM>
TPCH_SCALE_FACTOR 300g <SYSTEM>
TPCH_DATABASE tpch300g <SYSTEM>
TPCH_AUTOMATION_PATH C:\MSTPCH.205\setup\RF_Procs\CLRFs
<SYSTEM>
TPCH_AUTOMATION_CMD_PATH %TPCH_AUTOMATION_PATH%
<SYSTEM>
TPCH_AUTOMATION_SQL %TPCH_AUTOMATION_PATH%
<SYSTEM>
TPCH_AUTOMATION_CONFIG %TPCH_AUTOMATION_PATH%
<SYSTEM>
TPCH_AUTOMATION_EXEC %TPCH_AUTOMATION_PATH%
<SYSTEM>

TPCH_AUTOMATION_VBS %TPCH_AUTOMATION_PATH% <SYSTEM>
TPCH_AUTOMATION_MAPS %TPCH_AUTOMATION_PATH%\MAPS <SYSTEM>
TPCH_AUTOMATION_ROOT C:\TPCH_Automation_Kit <SYSTEM>
TPCH_AUTOMATION_SOURCES %TPCH_AUTOMATION_PATH%\Sources <SYSTEM>
TPCH_AUTOMATION_GENERATOR_SCRIPT_PATH %TPCH_AUTOMATION_PATH%\SQLScriptGenerators <SYSTEM>
TPCH_OUTPUT_DIR_PREFIX C:\tpchRuns <SYSTEM>
TPCH_MASTERLOG_NAME MasterLog.txt <SYSTEM>
TPCH_NUM_DATABASE_FILES 12 <SYSTEM>
TPCH_NUM_BACKUP_DEVICES 6 <SYSTEM>
TPCH_NUM_UPDATE_SETS 14 <SYSTEM>
TPCH_NUM_RF_LOAD_STREAMS 48 <SYSTEM>
TPCH_NUM_RF_LOAD_FILES 64 <SYSTEM>
TPCH_NUM_RF_EXEC_STREAMS 48 <SYSTEM>
TPCH_NUM_RF_EXEC_TOTAL 144 <SYSTEM>
TPCH_RF_EXEC_BATCH_SIZE 10 <SYSTEM>
TPCH_NUM_UPDATE_FILES_PER_SET %TPCH_NUM_RF_LOAD_FILES% <SYSTEM>
TPCH_NUM_TOTAL_FF_STREAMS 4 <SYSTEM>
TPCH_NUM_TOTAL_FF_FILES 16 <SYSTEM>
TPCH_NUM_TOTAL_FF_SETS 4 <SYSTEM>
TPCH_NUM_THROUGHPUT_STREAMS 6 <SYSTEM>
TPCH_NUM_LOAD_STREAMS 8 <SYSTEM>
TPCH_FF_DRIVES 1 <SYSTEM>
TPCH_DATA_DRIVES 12 <SYSTEM>
TPCH_LOG_DRIVES 1 <SYSTEM>
TPCH_BACKUP_DRIVES 6 <SYSTEM>
TPCH_TEMPDB_DATA_DRIVES 12 <SYSTEM>
TPCH_TEMPDB_LOG_DRIVES 1 <SYSTEM>
TPCH_FILE_GROWTH_SIZE 10 <SYSTEM>
TPCH_TEMPDEV_SIZE 30000MB <SYSTEM>
TPCH_TEMPLOG_SIZE 5000MB <SYSTEM>
DATA_FG_FILE_SIZE 50000MB <SYSTEM>
LOAD_FG_FILE_SIZE 50000MB <SYSTEM>
LOG_FG_FILE_SIZE 20000MB <SYSTEM>
TPCH_SQL_DIR %systemdrive%\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL <SYSTEM>
TPCH_SQL_LOG_DIR %systemdrive%\MyAutoMSSQL\Installed\MSSQL.1\MSSQL <SYSTEM>
TPCH_SQL_START_COMMAND_LINE -c -x -E -T834 -T1204 -T836 <SYSTEM>
TPCH_SQL_SHELL sqlcmd -Stpcha -E -I -b -w 65535 -l 60 -y 0 <SYSTEM>
TPCH_LOGMAN_COUNTERS_FILE %TPCH_AUTOMATION_CONFIG%\perfCountersTPCH.ini <SYSTEM>
TPCH_LOGMAN_SAMPLE_INTERVAL 1 <SYSTEM>
TPCH_LOGMAN_SAMPLE_FORMAT csv <SYSTEM>
TPCH_F1_INSTALL_PATH %TPCH_AUTOMATION_PATH%\installers <SYSTEM>
TPCH_F1_EXEC_DIR %systemdrive%\Program Files\F1Profiler <SYSTEM>
TPCH_KERNRATE_CMD -w -a -v 2 -b 16 -fs -ts -z sqlservr -z ntoskrnl -z hal -z PartMgr -z tcpip -z ftdisk -z scsiport -z hpqcissb -z hpqcissd -z hal -n sqlservr.exe <SYSTEM>
_NT_SYMBOL_PATH SRV*\\symbols\symbols <SYSTEM>
TPCH_EMON_COUNTER_LIST emonCounters_P3.txt <SYSTEM>
TPCH_BUDDY_BUILD_PATH <SYSTEM>
TPCH_EXEC_WINDOW_COLOR 9F <SYSTEM>
TEMP %USERPROFILE%\Local Settings\Temp NT AUTHORITY\SYSTEM
TMP %USERPROFILE%\Local Settings\Temp NT AUTHORITY\SYSTEM
TEMP %USERPROFILE%\Local Settings\Temp NT AUTHORITY\LOCAL SERVICE
TMP %USERPROFILE%\Local Settings\Temp NT AUTHORITY\LOCAL SERVICE
TEMP %USERPROFILE%\Local Settings\Temp NT AUTHORITY\NETWORK SERVICE
TMP %USERPROFILE%\Local Settings\Temp NT AUTHORITY\NETWORK SERVICE
TEMP %USERPROFILE%\Local Settings\Temp TPCHA\Administrator
TMP %USERPROFILE%\Local Settings\Temp TPCHA\Administrator
TPCH_SCALE_FACTOR_SIMPLE 300 TPCHA\Administrator
TPCH_SCALE_FACTOR 300g TPCHA\Administrator
TPCH_DATABASE tpch300g TPCHA\Administrator
TPCH_AUTOMATION_PATH C:\MSTPCH.205\setup\RF_Procs\CLRFs TPCHA\Administrator
TPCH_AUTOMATION_CMD_PATH %TPCH_AUTOMATION_PATH% TPCHA\Administrator
TPCH_AUTOMATION_SQL %TPCH_AUTOMATION_PATH% TPCHA\Administrator
TPCH_AUTOMATION_CONFIG %TPCH_AUTOMATION_PATH% TPCHA\Administrator
TPCH_AUTOMATION_EXEC %TPCH_AUTOMATION_PATH% TPCHA\Administrator
TPCH_AUTOMATION_VBS %TPCH_AUTOMATION_PATH% TPCHA\Administrator
TPCH_AUTOMATION_MAPS %TPCH_AUTOMATION_PATH%\MAPS TPCHA\Administrator
TPCH_AUTOMATION_ROOT C:\TPCH_Automation_Kit TPCHA\Administrator
TPCH_AUTOMATION_SOURCES %TPCH_AUTOMATION_PATH%\Sources TPCHA\Administrator
TPCH_AUTOMATION_GENERATOR_SCRIPT_PATH %TPCH_AUTOMATION_PATH%\SQLScriptGenerators TPCHA\Administrator
TPCH_OUTPUT_DIR_PREFIX C:\tpchRuns TPCHA\Administrator
TPCH_MASTERLOG_NAME MasterLog.txt TPCHA\Administrator
TPCH_NUM_DATABASE_FILES 12 TPCHA\Administrator
TPCH_NUM_BACKUP_DEVICES 6 TPCHA\Administrator
TPCH_NUM_UPDATE_SETS 14 TPCHA\Administrator
TPCH_NUM_RF_LOAD_STREAMS 64 TPCHA\Administrator
TPCH_NUM_RF_LOAD_FILES 64 TPCHA\Administrator
TPCH_NUM_RF_EXEC_STREAMS 48 TPCHA\Administrator
TPCH_NUM_RF_EXEC_TOTAL 48 TPCHA\Administrator
TPCH_RF_EXEC_BATCH_SIZE 10 TPCHA\Administrator
TPCH_NUM_UPDATE_FILES_PER_SET %TPCH_NUM_RF_LOAD_FILES% TPCHA\Administrator
TPCH_NUM_TOTAL_FF_STREAMS 4 TPCHA\Administrator
TPCH_NUM_TOTAL_FF_FILES 16 TPCHA\Administrator
TPCH_NUM_TOTAL_FF_SETS 4 TPCHA\Administrator
TPCH_NUM_THROUGHPUT_STREAMS 6 TPCHA\Administrator
TPCH_NUM_LOAD_STREAMS 8 TPCHA\Administrator
TPCH_FF_DRIVES 1 TPCHA\Administrator
TPCH_DATA_DRIVES 12 TPCHA\Administrator
TPCH_LOG_DRIVES 1 TPCHA\Administrator
TPCH_BACKUP_DRIVES 6 TPCHA\Administrator
TPCH_TEMPDB_DATA_DRIVES 12 TPCHA\Administrator
TPCH_TEMPDB_LOG_DRIVES 1 TPCHA\Administrator
TPCH_FILE_GROWTH_SIZE 10 TPCHA\Administrator
TPCH_TEMPDEV_SIZE 30000MB TPCHA\Administrator
TPCH_TEMPLOG_SIZE 5000MB TPCHA\Administrator
DATA_FG_FILE_SIZE 50000MB TPCHA\Administrator
LOAD_FG_FILE_SIZE 50000MB TPCHA\Administrator
LOG_FG_FILE_SIZE 20000MB TPCHA\Administrator
TPCH_SQL_DIR %systemdrive%\Program Files\Microsoft SQL Server\MSSQL.1\MSSQL TPCHA\Administrator
TPCH_SQL_LOG_DIR %systemdrive%\MyAutoMSSQL\Installed\MSSQL.1\MSSQL TPCHA\Administrator
TPCH_SQL_START_COMMAND_LINE -c -x -E -T834 -T2301 TPCHA\Administrator
TPCH_SQL_SHELL sqlcmd -Stpcha -E -I -b -w 65535 -l 60 -y 0 TPCHA\Administrator
TPCH_LOGMAN_COUNTERS_FILE %TPCH_AUTOMATION_CONFIG%\perfCountersTPCH.ini TPCHA\Administrator
TPCH_LOGMAN_SAMPLE_INTERVAL 1 TPCHA\Administrator
TPCH_LOGMAN_SAMPLE_FORMAT csv TPCHA\Administrator
TPCH_F1_INSTALL_PATH %TPCH_AUTOMATION_PATH%\installers TPCHA\Administrator
TPCH_F1_EXEC_DIR %systemdrive%\Program Files\F1Profiler TPCHA\Administrator
TPCH_KERNRATE_CMD -w -a -v 2 -b 16 -fs -ts -z sqlservr -z ntoskrnl -z hal -z PartMgr -z tcpip -z ftdisk -z scsiport -z hpqcissb -z hpqcissd -z hal -n sqlservr.exe TPCHA\Administrator
_NT_SYMBOL_PATH SRV*\\symbols\symbols TPCHA\Administrator

TPCH_EMON_COUNTER_LIST emonCounters_P3.txt
TPCHA\Administrator
TPCH_BUDDY_BUILD_PATH TPCHA\Administrator
TPCH_EXEC_WINDOW_COLOR 9F TPCHA\Administrator
PROCESSOR_IDENTIFIER=EM64T Family TPCHA\Administrator
TPCH_KERNRATE_CMD=-w -a TPCHA\Administrator
TPCH_SQL_SHELL=sqlcmd -Stpcha TPCHA\Administrator
TPCH_SQL_START_COMMAND_LINE=-c -x
TPCHA\Administrator
100g 300g TPCHA\Administrator

[Print Jobs]

Document Size Owner Notify Status Time Submitted Start
Time Until Time Elapsed Time Pages Printed Job ID Priority
Parameters Driver Print Processor Host Print Queue Data Type Name

[Network Connections]

Local Name Remote Name Type Status User Name
X: \\192.168.30.250\edrive Disk Persistent Connection
Y: \\192.168.30.250\ray Disk Persistent Connection
Z: \\192.168.30.250\edrive Disk Persistent Connection

[Running Tasks]

Name Path Process ID Priority Min Working Set Max Working Set
Start Time Version Size File Date
system idle process Not Available 0 0 Not Available
Not Available Not Available Not Available Not Available
Not Available
system Not Available 4 8 0 1380 Not
Available Not Available Not Available Not Available
smss.exe Not Available 380 11 200 1380
11/14/2007 8:39 AM Not Available Not Available Not Available
csrss.exe Not Available 428 13 Not Available Not
Available 11/14/2007 8:39 AM Not Available Not Available Not
Available
winlogon.exe c:\windows\system32\winlogon.exe 452 13
200 1380 11/14/2007 8:39 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 922.00 KB (944,128 bytes) 7/12/2007 1:13
PM
services.exe c:\windows\system32\services.exe 500 9
200 1380 11/14/2007 8:39 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 219.00 KB (224,256 bytes) 3/25/2005 7:00
AM
lsass.exe c:\windows\system32\lsass.exe 512 9 200 1380
11/14/2007 8:39 AM 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 14.00
KB (14,336 bytes) 3/25/2005 7:00 AM
svchost.exe c:\windows\system32\svchost.exe 700 8
200 1380 11/14/2007 8:39 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 25.00 KB (25,600 bytes) 7/12/2007 1:13
PM
svchost.exe Not Available 760 8 Not Available
Not Available 11/14/2007 8:39 AM Not Available Not Available
Not Available
svchost.exe Not Available 820 8 Not Available
Not Available 11/14/2007 8:39 AM Not Available Not Available
Not Available
svchost.exe c:\windows\system32\svchost.exe 848 8
200 1380 11/14/2007 8:39 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 25.00 KB (25,600 bytes) 7/12/2007 1:13
PM
msdtc.exe Not Available 936 8 Not Available Not
Available 11/14/2007 8:40 AM Not Available Not Available Not
Available
svchost.exe Not Available 1068 8 Not Available
Not Available 11/14/2007 8:40 AM Not Available Not Available
Not Available
vivaldiframework.exe c:\program files (x86)\megaraid storage
manager\framework\vivaldiframework.exe 1284 8 200 1380
11/14/2007 8:40 AM Not Available 52.00 KB (53,248 bytes)
11/29/2006 6:05 PM
mrmonitor.exe c:\program files (x86)\megaraid storage
manager\megamonitor\mrmonitor.exe 1532 8 200 1380
11/14/2007 8:40 AM Not Available 432.00 KB (442,368 bytes)
10/5/2007 12:59 PM
cmd.exe c:\windows\system32\cmd.exe 1528 8 200 1380
11/14/2007 8:40 AM 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 538.00
KB (550,912 bytes) 3/25/2005 7:00 AM
javaw.exe c:\program files (x86)\megaraid storage manager\jre\bin\javaw.exe 1548
8 200 1380 11/14/2007 8:40 AM 6.0.0.105 132.00 KB
(135,168 bytes) 9/11/2007 1:27 PM
svchost.exe c:\windows\system32\svchost.exe 1564 8
200 1380 11/14/2007 8:40 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 25.00 KB (25,600 bytes) 7/12/2007 1:13
PM
wmiprvse.exe Not Available 1812 8 Not Available
Not Available 11/14/2007 8:40 AM Not Available Not Available
Not Available
explorer.exe c:\windows\explorer.exe 1912 8 200
1380 11/14/2007 8:40 AM 6.00.3790.3959 (srv03_sp2_rtm.070216-1710)
1.30 MB (1,364,480 bytes) 7/12/2007 1:14 PM
popup.exe c:\program files (x86)\megaraid storage manager\megapopup\popup.exe
516 8 200 1380 11/14/2007 8:40 AM 1.00-24 76.10
KB (77,922 bytes) 8/8/2007 3:44 PM
svchost.exe c:\windows\system32\svchost.exe 920 8
200 1380 11/14/2007 8:40 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 25.00 KB (25,600 bytes) 7/12/2007 1:13
PM
csrss.exe Not Available 4216 13 Not Available Not
Available 11/14/2007 9:21 AM Not Available Not Available Not
Available
winlogon.exe c:\windows\system32\winlogon.exe 4244 13
200 1380 11/14/2007 9:21 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 922.00 KB (944,128 bytes) 7/12/2007 1:13
PM
rdpclip.exe c:\windows\system32\rdpclip.exe 4516 8 200 1380
11/14/2007 9:22 AM 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 99.00
KB (101,376 bytes) 7/11/2007 10:17 AM
explorer.exe c:\windows\explorer.exe 4648 8 200
1380 11/14/2007 9:22 AM 6.00.3790.3959 (srv03_sp2_rtm.070216-1710)
1.30 MB (1,364,480 bytes) 7/12/2007 1:14 PM
popup.exe c:\program files (x86)\megaraid storage manager\megapopup\popup.exe
1036 8 200 1380 11/14/2007 9:22 AM 1.00-24 76.10
KB (77,922 bytes) 8/8/2007 3:44 PM
cmd.exe c:\windows\system32\cmd.exe 7160 8 200 1380
11/14/2007 9:58 AM 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 538.00
KB (550,912 bytes) 3/25/2005 7:00 AM
sqlservr.exe c:\program files\microsoft sql
server\mssql.1\mssql\binn\sqlservr.exe 2996 8 200 1380
11/14/2007 10:04 AM 2005.090.3042.00 36.72 MB (38,507,376 bytes)
2/10/2007 9:03 AM
scrnsave.scr c:\windows\system32\scrnsave.scr 4352 4
200 1380 11/14/2007 10:28 AM 5.2.3790.1830
(srv03_sp1_rtm.050324-1447) 11.00 KB (11,264 bytes) 3/25/2005 7:00
AM
helpctr.exe c:\windows\pchealth\helpctr\binaries\helpctr.exe 5476 8
200 1380 11/14/2007 10:33 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 1.30 MB (1,363,456 bytes) 7/11/2007 10:20
AM
wmiprvse.exe Not Available 5272 8 Not Available
Not Available 11/14/2007 10:33 AM Not Available Not Available
Not Available
helpsvc.exe c:\windows\pchealth\helpctr\binaries\helpsvc.exe 5072
8 200 1380 11/14/2007 10:33 AM 5.2.3790.3959
(srv03_sp2_rtm.070216-1710) 1.52 MB (1,591,296 bytes) 7/11/2007 10:20
AM

[Loaded Modules]

Name Version Size File Date Manufacturer Path
winlogon 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 922.00 KB
(944,128 bytes) 7/12/2007 1:13 PM Microsoft Corporation
c:\windows\system32\winlogon.exe

ntdll 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.20 MB (1,254,400 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\ntdll.dll
kernel32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.43 MB (1,503,232 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\kernel32.dll
advapi32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.00 MB (1,051,648 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\advapi32.dll
rpcrt4 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.58 MB (1,653,248 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\rpcrt4.dll
secur32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 120.00 KB (122,880 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\secur32.dll
crypt32 5.131.3790.3959 (srv03_sp2_rtm.070216-1710) 1.36 MB (1,429,504 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\crypt32.dll
msvcrt 7.0.3790.3959 (srv03_sp2_rtm.070216-1710) 508.00 KB (520,192 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\msvcrt.dll
user32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.04 MB (1,086,976 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\user32.dll
gdi32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 603.50 KB (617,984 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\gdi32.dll
msasn1 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 152.50 KB (156,160 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\msasn1.dll
nddeapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 25.00 KB (25,600 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\nddeapi.dll
profmap 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 36.00 KB (36,864 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\profmap.dll
netapi32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 589.00 KB (603,136 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\netapi32.dll
userenv 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.02 MB (1,071,104 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\userenv.dll
psapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 29.00 KB (29,696 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\psapi.dll
regapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 108.50 KB (111,104 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\regapi.dll
setupapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.45 MB (1,524,224 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\setupapi.dll
version 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 28.00 KB (28,672 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\version.dll
winsta 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 89.00 KB (91,136 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\winsta.dll
ws2_32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 174.50 KB (178,688 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\ws2_32.dll
ws2help 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 30.50 KB (31,232 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\ws2help.dll
msgina 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.14 MB (1,193,472 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\msgina.dll
shsvcs 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 194.00 KB (198,656 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\shsvcs.dll
shlwapi 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 607.00 KB (621,568 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\shlwapi.dll
sfc 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 6.00 KB (6,144 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\sfc.dll
sfc_os 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 183.50 KB (187,904 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\sfc_os.dll
wintrust 5.131.3790.3959 (srv03_sp2_rtm.070216-1710) 297.50 KB (304,640 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wintrust.dll
imagehlp 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 57.50 KB (58,880 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\imagehlp.dll
ole32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 2.50 MB (2,622,976 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\ole32.dll
comctl32 6.0 (srv03_sp2_rtm.070216-1710) 1.51 MB (1,584,640 bytes) 2/18/2007 10:24 AM Microsoft Corporation c:\windows\winsxs\amd64_microsoft.windows.common-controls_6595b64144ccf1df_6.0.3790.3959_x-ww_0a7b2435\comctl32.dll
winscard 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 229.50 KB (235,008 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\winscard.dll
wtsapi32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 29.00 KB (29,696 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wtsapi32.dll
sxs 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.89 MB (1,977,856 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\sxs.dll
shell32 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 10.02 MB (10,505,728 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\shell32.dll
wldap32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 390.50 KB (399,872 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wldap32.dll
rsaenh 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 298.84 KB (306,008 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\rsaenh.dll
cscdll 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 151.50 KB (155,136 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\cscdll.dll
dimsntfy 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 28.00 KB (28,672 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\dimsntfy.dll
wlnotify 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 148.50 KB (152,064 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wlnotify.dll
winmm 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 303.50 KB (310,784 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\winmm.dll
winspool 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 247.00 KB (252,928 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\winspool.drv
mpr 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 115.00 KB (117,760 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\mpr.dll
oleaut32 5.2.3790.3959 1.07 MB (1,121,792 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\oleaut32.dll
comctl32 5.82 (srv03_sp2_rtm.070216-1710) 935.00 KB (957,440 bytes) 2/18/2007 10:24 AM Microsoft Corporation c:\windows\winsxs\amd64_microsoft.windows.common-controls_6595b64144ccf1df_5.82.3790.3959_x-ww_ab06deb0\comctl32.dll
uxtheme 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 494.50 KB (506,368 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\uxtheme.dll
samlib 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 69.50 KB (71,168 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\samlib.dll
clbcatq 2001.12.4720.3959 (srv03_sp2_rtm.070216-1710) 862.50 KB (883,200 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\clbcatq.dll
comres 2001.12.4720.3959 (srv03_sp2_rtm.070216-1710) 779.50 KB (798,208 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\comres.dll

xpsp2res 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 2.77 MB (2,899,456 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\xpsp2res.dll
cscui 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 441.00 KB (451,584 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\cscui.dll
mprapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 154.50 KB (158,208 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\mprapi.dll
activeds 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 348.50 KB (356,864 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\activeds.dll
adsldpc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 240.50 KB (246,272 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\adsldpc.dll
credui 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 202.00 KB (206,848 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\credui.dll
atl 3.05.2284 96.50 KB (98,816 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\atl.dll
rtutils 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 66.00 KB (67,584 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\rtutils.dll
drprov 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 24.00 KB (24,576 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\drprov.dll
ntlanman 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 72.00 KB (73,728 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\ntlanman.dll
netui0 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 130.00 KB (133,120 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\netui0.dll
netui1 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 338.50 KB (346,624 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\netui1.dll
davclnt 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 39.50 KB (40,448 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\davclnt.dll
mprui 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 67.50 KB (69,120 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\mprui.dll
netui2 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 542.00 KB (555,008 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\netui2.dll
comdlg32 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 447.00 KB (457,728 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\comdlg32.dll
netmsg 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 179.00 KB (183,296 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\netmsg.dll
ntmarta 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 222.50 KB (227,840 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\ntmarta.dll
wbemprox 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 38.00 KB (38,912 bytes) 7/11/2007 10:17 AM Microsoft Corporation c:\windows\system32\wbem\wbemprox.dll
wbemcomn 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 518.00 KB (530,432 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\wbemcomn.dll
wbemsvc 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 58.00 KB (59,392 bytes) 7/11/2007 10:17 AM Microsoft Corporation c:\windows\system32\wbem\wbemsvc.dll
fastprox 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 866.50 KB (887,296 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\fastprox.dll
msvcp60 7.0.3790.1830 (srv03_sp1_rtm.050324-1447) 919.50 KB (941,568 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msvcp60.dll
ntdsapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 127.50 KB (130,560 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\ntdsapi.dll
dnsapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 298.50 KB (305,664 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\dnsapi.dll
services 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 219.00 KB (224,256 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\services.exe
scesrv 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 595.00 KB (609,280 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\scesrv.dll
authz 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 171.00 KB (175,104 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\authz.dll
umpnpmgr 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 205.00 KB (209,920 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\umpnpmgr.dll
ncobjapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 77.50 KB (79,360 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\ncobjapi.dll
eventlog 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 127.50 KB (130,560 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\eventlog.dll
cryptnet 5.131.3790.3959 (srv03_sp2_rtm.070216-1710) 108.50 KB (111,104 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\cryptnet.dll
sensapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 10.50 KB (10,752 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\sensapi.dll
cabinet 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 138.50 KB (141,824 bytes) 2/18/2007 9:50 AM Microsoft Corporation c:\windows\system32\cabinet.dll
imm32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 208.00 KB (212,992 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\imm32.dll
apphelp 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 241.00 KB (246,784 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\apphelp.dll
lsass 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 14.00 KB (14,336 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\lsass.exe
lsasrv 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.49 MB (1,566,720 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\lsasrv.dll
samsrv 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.01 MB (1,059,328 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\samsrv.dll
cryptdll 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 47.00 KB (48,128 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\cryptdll.dll
msprivs 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 47.50 KB (48,640 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msprivs.dll
kerberos 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 706.00 KB (722,944 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\kerberos.dll
msv1_0 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 253.00 KB (259,072 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\msv1_0.dll
iphlpapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 178.50 KB (182,784 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\iphlpapi.dll
netlogon 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 665.50 KB (681,472 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\netlogon.dll
w32time 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 398.00 KB (407,552 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\w32time.dll
schannel 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 248.00 KB (253,952 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\schannel.dll
wdigest 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 131.00 KB (134,144 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wdigest.dll
rassfm 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 36.00 KB (36,864 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\rassfm.dll

kdcsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 410.00 KB (419,840 bytes) 7/12/2007 1:15 PM Microsoft Corporation c:\windows\system32\kdcsvc.dll
ntdsa 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 2.83 MB (2,967,040 bytes) 7/12/2007 1:15 PM Microsoft Corporation c:\windows\system32\ntdsa.dll
ntdsatq 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 51.00 KB (52,224 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\ntdsatq.dll
mswsock 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 480.50 KB (492,032 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\mswsock.dll
esent 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 2.26 MB (2,367,488 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\esent.dll
scecli 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 308.00 KB (315,392 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\scecli.dll
ws03res 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 794.00 KB (813,056 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\ws03res.dll
pstorsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 36.00 KB (36,864 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\pstorsvc.dll
psbase 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 124.00 KB (126,976 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\psbase.dll
hnetcfg 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 568.00 KB (581,632 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\hnetcfg.dll
wshtcpip 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 29.00 KB (29,696 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wshtcpip.dll
dssenh 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 231.34 KB (236,888 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\dssenh.dll
svchost 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 25.00 KB (25,600 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\svchost.exe
rpcss 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 819.00 KB (838,656 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\rpcss.dll
schedsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 308.00 KB (315,392 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\schedsvc.dll
msidle 6.00.3790.1830 (srv03_sp1_rtm.050324-1447) 9.00 KB (9,216 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msidle.dll
wkssvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 221.00 KB (226,304 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\wkssvc.dll
wiarpc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 57.00 KB (58,368 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wiarpc.dll
dmserver 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 36.50 KB (37,376 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\dmserver.dll
es 2001.12.4720.3959 (srv03_sp2_rtm.070216-1710) 357.00 KB (365,568 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\es.dll
pchsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 75.50 KB (77,312 bytes) 7/11/2007 10:20 AM Microsoft Corporation c:\windows\pchealth\helpctr\binaries\pchsvc.dll
hidserv 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 24.00 KB (24,576 bytes) 7/11/2007 6:04 AM Microsoft Corporation c:\windows\system32\hidserv.dll
hid 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 33.00 KB (33,792 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\hid.dll
srvsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 156.50 KB (160,256 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\srvsvc.dll
seclogon 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 27.50 KB (28,160 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\seclogon.dll
trkwks 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 177.50 KB (181,760 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\trkwks.dll
wmisvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 227.50 KB (232,960 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\wmisvc.dll
vssapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.26 MB (1,320,960 bytes) 7/12/2007 1:15 PM Microsoft Corporation c:\windows\system32\vssapi.dll
sens 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 63.50 KB (65,024 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\sens.dll
comsvcs 2001.12.4720.3959 (srv03_sp2_rtm.070216-1710) 2.13 MB (2,234,880 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\comsvcs.dll
wbemcore 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.22 MB (1,282,560 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\wbemcore.dll
esscli 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 626.50 KB (641,536 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\esscli.dll
wmiutils 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 171.00 KB (175,104 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\wmiutils.dll
repdrvfs 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 353.50 KB (361,984 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\repdrvfs.dll
wmiprvsd 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 743.00 KB (760,832 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\wmiprvsd.dll
wbemess 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 532.50 KB (545,280 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\wbem\wbemess.dll
ncprov 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 73.00 KB (74,752 bytes) 7/11/2007 10:17 AM Microsoft Corporation c:\windows\system32\wbem\ncprov.dll
netman 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 454.50 KB (465,408 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\netman.dll
netshell 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 2.33 MB (2,438,656 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\netshell.dll
clusapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 127.00 KB (130,048 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\clusapi.dll
rasapi32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 410.00 KB (419,840 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\rasapi32.dll
rasman 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 95.50 KB (97,792 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\rasman.dll
tapi32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 332.50 KB (340,480 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\tapi32.dll
wzcsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 644.50 KB (659,968 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wzcsvc.dll
wmi 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 5.50 KB (5,632 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\wmi.dll
dhcpcsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 220.50 KB (225,792 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\dhcpcsvc.dll
wininet 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 1.14 MB (1,190,912 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wininet.dll
wzcsapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 50.50 KB (51,712 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wzcsapi.dll

netcfgx 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.29 MB (1,354,752 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\netcfgx.dll
winipsec 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 52.50 KB (53,760 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\winipsec.dll
rasmans 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 281.00 KB (287,744 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\rasmans.dll
rastapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 83.50 KB (85,504 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\rastapi.dll
rasppp 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 297.00 KB (304,128 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\rasppp.dll
ntlsapi 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 11.00 KB (11,264 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\ntlsapi.dll
raschap 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 141.00 KB (144,384 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\raschap.dll
rastls 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 238.50 KB (244,224 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\rastls.dll
cryptui 5.131.3790.3959 (srv03_sp2_rtm.070216-1710) 705.50 KB (722,432 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\cryptui.dll
ipbootp 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 47.50 KB (48,640 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\ipbootp.dll
rasdlg 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 859.50 KB (880,128 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\rasdlg.dll
rasadhlp 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 12.00 KB (12,288 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\rasadhlp.dll
cryptsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 114.00 KB (116,736 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\cryptsvc.dll
certcli 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 371.50 KB (380,416 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\certcli.dll
VivaldiFramework Not Available 52.00 KB (53,248 bytes) 11/29/2006 6:05 PM Not Available c:\program files (x86)\megaraid storage manager\framework\vivaldiframework.exe
wow64 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 244.50 KB (250,368 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wow64.dll
wow64win 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 280.00 KB (286,720 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\wow64win.dll
wow64cpu 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 18.50 KB (18,944 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\wow64cpu.dll
mrmonitor Not Available 432.00 KB (442,368 bytes) 10/5/2007 12:59 PM Not Available c:\program files (x86)\megaraid storage manager\megamonitor\mrmonitor.exe
cmd 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 538.00 KB (550,912 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\cmd.exe
javaw 6.0.0.105 132.00 KB (135,168 bytes) 9/11/2007 1:27 PM Sun Microsystems, Inc. c:\program files (x86)\megaraid storage manager\jre\bin\javaw.exe
termsrv 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 355.50 KB (364,032 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\termsrv.dll
icaapi 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 27.50 KB (28,160 bytes) 7/11/2007 10:17 AM Microsoft Corporation c:\windows\system32\icaapi.dll
mstlsapi 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 188.00 KB (192,512 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\mstlsapi.dll
rdpwsx 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 170.13 KB (174,216 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\rdpwsx.dll
explorer 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 1.30 MB (1,364,480 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\explorer.exe
browseui 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 1.53 MB (1,605,120 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\browseui.dll
shdocvw 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 2.33 MB (2,438,144 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\shdocvw.dll
themeui 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 531.50 KB (544,256 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\themeui.dll
msimg32 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 6.50 KB (6,656 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msimg32.dll
linkinfo 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 31.00 KB (31,744 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\linkinfo.dll
ntshrui 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 184.00 KB (188,416 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\ntshrui.dll
webcheck 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 438.50 KB (449,024 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\webcheck.dll
wsock32 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 24.50 KB (25,088 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\wsock32.dll
stobject 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 142.50 KB (145,920 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\stobject.dll
batmeter 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 41.50 KB (42,496 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\batmeter.dll
powrprof 6.00.3790.1830 (srv03_sp1_rtm.050324-1447) 32.50 KB (33,280 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\powrprof.dll
urlmon 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 1.04 MB (1,088,000 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\urlmon.dll
browselc 6.00.3790.1830 (srv03_sp1_rtm.050324-1447) 63.00 KB (64,512 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\browselc.dll
shdoclc 6.00.3790.1830 (srv03_sp1_rtm.050324-1447) 589.50 KB (603,648 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\shdoclc.dll
mydocs 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 101.00 KB (103,424 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\mydocs.dll
jscript 5.6.0.8832 976.00 KB (999,424 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\jscript.dll
winrnr 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 30.00 KB (30,720 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\winrnr.dll
dfshim 2.0.50727.42 (RTM.050727-4200) 95.00 KB (97,280 bytes) 9/22/2005 11:52 PM Microsoft Corporation c:\windows\system32\dfshim.dll
mscoree 2.0.50727.42 (RTM.050727-4200) 441.00 KB (451,584 bytes) 9/22/2005 11:37 PM Microsoft Corporation c:\windows\system32\mscoree.dll
mlang 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 686.00 KB (702,464 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\mlang.dll
shfusion 2.0.50727.42 (RTM.050727-4200) 105.50 KB (108,032 bytes) 9/22/2005 11:36 PM Microsoft Corporation c:\windows\microsoft.net\framework64\v2.0.50727\shfusion.dll
msvcr80 8.00.50727.42 803.50 KB (822,784 bytes) 9/22/2005 11:26 PM Microsoft Corporation c:\windows\winsxs\amd64_microsoft.vc80.crt_1fc8b3b9a1e18e3b_8.0.50727.42_x-ww_3fea50ad\msvcr80.dll

fusion 2.0.50727.42 (RTM.050727-4200) 11.00 KB (11,264 bytes) 9/22/2005 11:37 PM Microsoft Corporation c:\windows\microsoft.net\framework64\v2.0.50727\fusion.dll
culture 2.0.50727.42 (RTM.050727-4200) 22.00 KB (22,528 bytes) 9/22/2005 11:41 PM Microsoft Corporation c:\windows\microsoft.net\framework64\v2.0.50727\culture.dll
shfusres 2.0.50727.42 (RTM.050727-4200) 85.00 KB (87,040 bytes) 9/22/2005 11:36 PM Microsoft Corporation c:\windows\microsoft.net\framework64\v2.0.50727\shfusres.dll
mmcshext 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 88.00 KB (90,112 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\mmcshext.dll
hhsetup 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 78.00 KB (79,872 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\hhsetup.dll
popup 1.00-24 76.10 KB (77,922 bytes) 8/8/2007 3:44 PM c:\program files (x86)\megaraid storage manager\megapopup\popup.exe
tapisrv 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 398.00 KB (407,552 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\tapisrv.dll
unimdm 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 302.00 KB (309,248 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\unimdm.tsp
uniplat 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 18.50 KB (18,944 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\uniplat.dll
kmddsp 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 56.50 KB (57,856 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\kmddsp.tsp
ndptsp 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 76.00 KB (77,824 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\ndptsp.tsp
ipconf 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 22.50 KB (23,040 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\ipconf.tsp
h323 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 424.00 KB (434,176 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\h323.tsp
hidphone 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 48.00 KB (49,152 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\hidphone.tsp
rdpsnd 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 25.00 KB (25,600 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\rdpsnd.dll
scredir 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 38.50 KB (39,424 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\scredir.dll
rdpclip 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 99.00 KB (101,376 bytes) 7/11/2007 10:17 AM Microsoft Corporation c:\windows\system32\rdpclip.exe
msacm32 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 112.00 KB (114,688 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msacm32.dll
actxprxy 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 220.50 KB (225,792 bytes) 7/12/2007 1:14 PM Microsoft Corporation c:\windows\system32\actxprxy.dll
zipfldr 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 449.50 KB (460,288 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\zipfldr.dll
sqlservr 2005.090.3042.00 36.72 MB (38,507,376 bytes) 2/10/2007 9:03 AM Microsoft Corporation c:\program files\microsoft sql server\mssql.1\mssql\binn\sqlservr.exe
msvcp80 8.00.50727.42 1.05 MB (1,097,728 bytes) 9/22/2005 11:28 PM Microsoft Corporation c:\windows\winsxs\amd64_microsoft.vc80.crt_1fc8b3b9a1e18e3b_8.0.50727.42_x-ww_3fea50ad\msvcp80.dll
opends60 2005.090.1399.00 22.21 KB (22,744 bytes) 10/14/2005 2:31 PM Microsoft Corporation c:\program files\microsoft sql server\mssql.1\mssql\binn\opends60.dll
instapi 2005.090.1399.00 40.71 KB (41,688 bytes) 10/14/2005 2:23 PM Microsoft Corporation c:\program files\microsoft sql server\90\shared\instapi.dll
sqlevn70 2005.090.3042.00 1.66 MB (1,740,656 bytes) 2/10/2007 9:02 AM Microsoft Corporation c:\program files\microsoft sql server\mssql.1\mssql\binn\resources\1033\sqlevn70.rll
sqlos 2005.090.3042.00 17.86 KB (18,288 bytes) 2/10/2007 9:03 AM Microsoft Corporation c:\program files\microsoft sql server\mssql.1\mssql\binn\sqlos.dll
xolehlp 2001.12.4720.3959 (srv03_sp2_rtm.070216-1710) 10.50 KB (10,752 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\xolehlp.dll
msdtcprx 2001.12.4720.3959 (srv03_sp2_rtm.070216-1710) 814.00 KB (833,536 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\msdtcprx.dll
mtxclu 2001.12.4720.3959 (srv03_sp2_rtm.070216-1710) 142.00 KB (145,408 bytes) 7/12/2007 1:13 PM Microsoft Corporation c:\windows\system32\mtxclu.dll
resutils 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 98.50 KB (100,864 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\resutils.dll
security 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 6.00 KB (6,144 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\security.dll
msfte 12.0.6828.0 3.63 MB (3,804,952 bytes) 8/28/2006 4:17 AM Microsoft Corporation c:\program files\microsoft sql server\mssql.1\mssql\binn\msfte.dll
dbghelp 6.6.0007.5 (debuggers(dbg).051022-1733) 1.27 MB (1,329,520 bytes) 2/10/2007 8:56 AM Microsoft Corporation c:\program files\microsoft sql server\90\shared\dbghelp.dll
sqlncli 2005.090.3042.00 2.74 MB (2,868,592 bytes) 2/10/2007 9:03 AM Microsoft Corporation c:\windows\system32\sqlncli.dll
sqlnclir 2005.090.1399.00 201.21 KB (206,040 bytes) 10/14/2005 2:31 PM Microsoft Corporation c:\windows\system32\sqlnclir.rll
scrnsave 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 11.00 KB (11,264 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\scrnsave.scr
helpctr 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.30 MB (1,363,456 bytes) 7/11/2007 10:20 AM Microsoft Corporation c:\windows\pchealth\helpctr\binaries\helpctr.exe
hcappres 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 7.50 KB (7,680 bytes) 7/11/2007 10:20 AM Microsoft Corporation c:\windows\pchealth\helpctr\binaries\hcappres.dll
itss 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 208.50 KB (213,504 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\itss.dll
msxml3 8.80.1185.0 2.04 MB (2,144,256 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msxml3.dll
pchshell 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 155.00 KB (158,720 bytes) 7/11/2007 10:20 AM Microsoft Corporation c:\windows\pchealth\helpctr\binaries\pchshell.dll
mshtml 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 5.72 MB (5,999,616 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\mshtml.dll
msls31 3.10.349.0 357.00 KB (365,568 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msls31.dll
msimtf 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 380.50 KB (389,632 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msimtf.dll
msctf 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 617.50 KB (632,320 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\msctf.dll
mshtmled 6.00.3790.3959 (srv03_sp2_rtm.070216-1710) 905.50 KB (927,232 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\mshtmled.dll
vbscript 5.6.0.8832 647.00 KB (662,528 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\vbscript.dll
msinfo 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 635.50 KB (650,752 bytes) 7/11/2007 10:20 AM Microsoft Corporation c:\windows\pchealth\helpctr\binaries\msinfo.dll
mfc42u 6.50.9146.0 1.39 MB (1,460,992 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\mfc42u.dll
riched32 5.2.3790.1830 (srv03_sp1_rtm.050324-1447) 7.00 KB (7,168 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\riched32.dll
riched20 5.31.23.1225 1.11 MB (1,160,192 bytes) 3/25/2005 7:00 AM Microsoft Corporation c:\windows\system32\riched20.dll

helpsvc 5.2.3790.3959 (srv03_sp2_rtm.070216-1710) 1.52 MB
(1,591,296 bytes) 7/11/2007 10:20 AM Microsoft Corporation
c:\windows\pchealth\helpctr\binaries\helpsvc.exe

[Services]

Display Name Name State Start ModeService Type Path
Error Control Start NameTag ID
Application Experience Lookup Service AeLookupSvc Stopped
Disabled Share Process c:\windows\system32\svchost.exe -k netsvcs Normal
LocalSystem 0
Alerter Alerter Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k localservice Normal NT
AUTHORITY\LocalService 0
Application Layer Gateway Service ALG Stopped Manual Own
Process c:\windows\system32\alg.exe Normal NT
AUTHORITY\LocalService 0
Application Management AppMgmt Stopped Manual Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
ASP.NET State Service aspnet_state Stopped Manual Own
Process c:\windows\microsoft.net\framework64\v2.0.50727\aspnet_state.exe
Normal NT AUTHORITY\NetworkService 0
Windows Audio AudioSrv Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Background Intelligent Transfer Service BITS Stopped Manual Share
Process c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem
0
Computer Browser Browser Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Indexing Service CiSvc Stopped Disabled Share Process
c:\windows\system32\cisvc.exe Normal LocalSystem 0
ClipBook ClipSrv Stopped Disabled Own Process
c:\windows\system32\clipsrv.exe Normal LocalSystem 0
.NET Runtime Optimization Service v2.0.50727_X86
clr_optimization_v2.0.50727_32 Stopped Manual Own Process
c:\windows\microsoft.net\framework\v2.0.50727\mscorsvw.exe Ignore
LocalSystem 0
.NET Runtime Optimization Service v2.0.50727_x64
clr_optimization_v2.0.50727_64 Stopped Manual Own Process
c:\windows\microsoft.net\framework64\v2.0.50727\mscorsvw.exe Ignore
LocalSystem 0
COM+ System Application COMSysApp Stopped Manual Own
Process c:\windows\system32\dllhost.exe
/processid:{02d4b3f1-fd88-11d1-960d-00805fc79235} Normal LocalSystem
0
Cryptographic Services CryptSvc Running Manual Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
DCOM Server Process Launcher DcomLaunch Running Auto Share
Process c:\windows\system32\svchost.exe -k dcomlaunch Normal
LocalSystem 0
Distributed File System Dfs Stopped Manual Own Process
c:\windows\system32\dfssvc.exe Normal LocalSystem 0
DHCP Client Dhcp Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k networkservice Normal NT
AUTHORITY\NetworkService 0
Logical Disk Manager Administrative Service dmadmin Stopped Manual
Share Process c:\windows\system32\dmadmin.exe /com Normal
LocalSystem 0
Logical Disk Managerdmserver Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
DNS Client Dnscache Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k networkservice Normal NT
AUTHORITY\NetworkService 0
Error Reporting Service ERSvc Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k winerr Ignore LocalSystem 0
Event Log Eventlog Running Auto Share Process
c:\windows\system32\services.exe Normal LocalSystem 0
COM+ Event System EventSystem Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Help and Support helpsvc Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
HID Input Service HidServ Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
HTTP SSL HTTPFilter Stopped Manual Share Process
c:\windows\system32\lsass.exe Normal LocalSystem 0
IAS Jet Database Access IASJet Stopped Manual Share Process
c:\windows\syswow64\svchost.exe -k iasjet Normal LocalSystem 0
InstallDriver Table Manager IDriverT Stopped Manual Own Process
"c:\program files (x86)\common files\installshield\driver\1050\intel 32\idrivert.exe"
Ignore LocalSystem 0
IMAPI CD-Burning COM Service ImapiService Stopped
Disabled Own Process c:\windows\system32\imapi.exe Normal
LocalSystem 0
Intersite Messaging IsmServ Stopped Disabled Own Process
c:\windows\system32\ismserv.exe Normal LocalSystem 0
Kerberos Key Distribution Center kdc Stopped Disabled Share
Process c:\windows\system32\lsass.exe Normal LocalSystem 0
Server lanmanserver Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Workstation lanmanworkstation Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
License Logging LicenseService Stopped Disabled Own Process
c:\windows\system32\llssrv.exe Normal NT AUTHORITY\NetworkService
0
TCP/IP NetBIOS Helper LmHosts Running Auto Share Process
c:\windows\system32\svchost.exe -k localservice Normal NT
AUTHORITY\LocalService 0
MRMonitor MegaMonitorSrv Running Auto Own Process
"c:\program files (x86)\megaraid storage manager\megamonitor\mrmonitor.exe"
Normal LocalSystem 0
Messenger Messenger Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
NetMeeting Remote Desktop Sharing mnmsrvc Stopped Disabled Own
Process c:\windows\system32\mnmsrvc.exe Normal LocalSystem
0
Distributed Transaction Coordinator MSDTC Running Auto Own
Process c:\windows\system32\msdtc.exe Normal NT
AUTHORITY\NetworkService 0
SQL Server FullText Search (MSSQLSERVER) msftesql Stopped
Disabled Own Process "c:\program files\microsoft sql
server\mssql.1\mssql\binn\msftesql.exe" -s:mssql.1 -f:mssqlserver Normal
LocalSystem 0
Windows Installer MSIServer Stopped Manual Share Process
c:\windows\system32\msiexec.exe /v Normal LocalSystem 0
MSMFramework MSMFramework Running Auto Own Process
"c:\program files (x86)\megaraid storage manager\framework\vivaldiframework.exe"
Normal LocalSystem 0
SQL Server (MSSQLSERVER) MSSQLSERVER Stopped Manual Own
Process "c:\program files\microsoft sql server\mssql.1\mssql\binn\sqlservr.exe"
-smssqlserver Normal LocalSystem 0
SQL Server Active Directory Helper MSSQLServerADHelper
Stopped Disabled Own Process "c:\program files\microsoft sql
server\90\shared\sqladhlp90.exe" Normal NT AUTHORITY\NetworkService
0
SQL Server Analysis Services (MSSQLSERVER) MSSQLServerOLAPService
Stopped Manual Own Process "c:\program files\microsoft sql
server\mssql.2\olap\bin\msmdsrv.exe" -s "c:\program files\microsoft sql
server\mssql.2\olap\config" Normal LocalSystem 0
Network DDE NetDDE Stopped Disabled Share Process
c:\windows\system32\netdde.exe Normal LocalSystem 0
Network DDE DSDM NetDDEdsdm Stopped Disabled Share Process
c:\windows\system32\netdde.exe Normal LocalSystem 0
Net Logon Netlogon Stopped Manual Share Process
c:\windows\system32\lsass.exe Normal LocalSystem 0
Network Connections Netman Running Manual Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Network Location Awareness (NLA) Nla Running Manual Share
Process c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem
0
File Replication NtFrs Stopped Manual Own Process
c:\windows\system32\ntfrs.exe Ignore LocalSystem 0
NT LM Security Support Provider NtLmSsp Stopped Manual Share
Process c:\windows\system32\lsass.exe Normal LocalSystem 0
Removable Storage NtmsSvc Stopped Manual Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0

Office Source Engine ose Stopped Manual Own Process
"c:\program files (x86)\common files\microsoft shared\source engine\ose.exe" Normal
LocalSystem 0
Plug and Play PlugPlay Running Auto Share Process
c:\windows\system32\services.exe Normal LocalSystem 0
IPSEC Services PolicyAgent Stopped Disabled Share Process
c:\windows\system32\lsass.exe Normal LocalSystem 0
Protected Storage ProtectedStorage Running Auto Share Process
c:\windows\system32\lsass.exe Normal LocalSystem 0
Remote Access Auto Connection Manager RasAuto Stopped Manual Share
Process c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem
0
Remote Access Connection Manager RasMan Running Manual Share
Process c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem
0
Remote Desktop Help Session Manager RDSessMgr Stopped Manual
Own Process c:\windows\system32\sessmgr.exe Normal
LocalSystem 0
Routing and Remote Access RemoteAccess Stopped Disabled Share
Process c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem
0
Remote Registry RemoteRegistry Running Auto Share Process
c:\windows\system32\svchost.exe -k regsvc Normal NT
AUTHORITY\LocalService 0
Remote Procedure Call (RPC) Locator RpcLocator Stopped Manual
Own Process c:\windows\system32\locator.exe Normal NT
AUTHORITY\NetworkService 0
Remote Procedure Call (RPC) RpcSs Running Auto Share Process
c:\windows\system32\svchost.exe -k rpcss Normal NT Authority\NetworkService
0
Resultant Set of Policy Provider RSoPProv Stopped Manual Share Process
c:\windows\system32\rsopprov.exe Normal LocalSystem 0
Special Administration Console Helper sacsvr Stopped Manual Share
Process c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem
0
Security Accounts Manager SamSs Running Auto Share Process
c:\windows\system32\lsass.exe Normal LocalSystem 0
Smart Card SCardSvr Stopped Manual Share Process
c:\windows\system32\scardsvr.exe Ignore NT
AUTHORITY\LocalService 0
Task Scheduler Schedule Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Secondary Logon seclogon Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Ignore LocalSystem 0
System Event Notification SENS Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Windows Firewall/Internet Connection Sharing (ICS) SharedAccess
Stopped Disabled Share Process c:\windows\system32\svchost.exe -k
netsvcs Normal LocalSystem 0
Shell Hardware Detection ShellHWDetection Running Auto Share
Process c:\windows\system32\svchost.exe -k netsvcs Ignore LocalSystem
0
Print Spooler Spooler Stopped Disabled Own Process
c:\windows\system32\spoolsv.exe Normal LocalSystem 0
SQL Server Browser SQLBrowser Stopped Disabled Own Process
"c:\program files (x86)\microsoft sql server\90\shared\sqlbrowser.exe" Normal
LocalSystem 0
SQL Server Agent (MSSQLSERVER) SQLSERVERAGENT
Stopped Disabled Own Process "c:\program files\microsoft sql
server\mssql.1\mssql\binn\sqlagent90.exe" -i mssqlserver Normal
LocalSystem 0
SQL Server VSS Writer SQLWriterStopped Disabled Own Process
"c:\program files\microsoft sql server\90\shared\sqlwriter.exe" Normal
LocalSystem 0
Windows Image Acquisition (WIA) stisvc Stopped Disabled Share
Process c:\windows\system32\svchost.exe -k imgsvc Normal NT
AUTHORITY\LocalService 0
Microsoft Software Shadow Copy Provider swprv Stopped Manual Own
Process c:\windows\system32\svchost.exe -k swprv Normal LocalSystem
0
Performance Logs and Alerts SysmonLog Stopped Auto Own
Process c:\windows\system32\smlogsvc.exe Normal NT
Authority\NetworkService 0
Telephony TapiSrv Running Manual Share Process
c:\windows\system32\svchost.exe -k tapisrv Normal LocalSystem 0
Terminal Services TermService Running Manual Share Process
c:\windows\system32\svchost.exe -k termsvcs Normal LocalSystem
0
Themes Themes Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Telnet TlntSvr Stopped Disabled Own Process
c:\windows\system32\tlntsvr.exe Normal NT AUTHORITY\LocalService 0
Distributed Link Tracking Server TrkSvr Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Distributed Link Tracking Client TrkWks Running Auto Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Terminal Services Session Directory Tssdis Stopped Disabled Own
Process c:\windows\system32\tssdis.exe Normal LocalSystem 0
Windows User Mode Driver Framework UMWdf Stopped Manual Own
Process c:\windows\system32\wdfmgr.exe Normal NT
AUTHORITY\LocalService 0
Uninterruptible Power Supply UPS Stopped Manual Own Process
c:\windows\system32\ups.exe Normal NT AUTHORITY\LocalService 0
Virtual Disk Service vds Stopped Manual Own Process
c:\windows\system32\vds.exe Normal LocalSystem 0
Volume Shadow Copy VSS Stopped Manual Own Process
c:\windows\system32\vssvc.exe Normal LocalSystem 0
Windows Time W32Time Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k localservice Normal NT
AUTHORITY\LocalService 0
WebClient WebClient Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k localservice Normal NT
AUTHORITY\LocalService 0
WinHTTP Web Proxy Auto-Discovery Service WinHttpAutoProxySvc
Stopped Manual Share Process c:\windows\system32\svchost.exe -k
localservice Normal NT AUTHORITY\LocalService 0
Windows Management Instrumentation winmgmt Running Auto Share
Process c:\windows\system32\svchost.exe -k netsvcs Ignore LocalSystem
0
Portable Media Serial Number Service WmdmPmSN Stopped Manual
Share Process c:\windows\system32\svchost.exe -k netsvcs Normal
LocalSystem 0
Windows Management Instrumentation Driver Extensions Wmi
Stopped Manual Share Process c:\windows\system32\svchost.exe -k
netsvcs Normal LocalSystem 0
WMI Performance Adapter WmiApSrv Stopped Manual Own
Process c:\windows\system32\wbem\wmiapsrv.exe Normal LocalSystem
0
Automatic Updates wuauserv Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Wireless Configuration WZCSVC Stopped Disabled Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0
Network Provisioning Service xmlprov Stopped Manual Share Process
c:\windows\system32\svchost.exe -k netsvcs Normal LocalSystem 0

[Program Groups]

Group Name Name User Name
Accessories Default User:Accessories Default User
Accessories\Accessibility Default User:Accessories\Accessibility Default
User
Accessories\Entertainment Default User:Accessories\Entertainment Default
User
Startup Default User:Startup Default User
Accessories All Users:Accessories All Users
Accessories\Accessibility All Users:Accessories\Accessibility All
Users
Accessories\Communications All Users:Accessories\Communications All
Users
Accessories\Entertainment All Users:Accessories\Entertainment All
Users
Accessories\System Tools All Users:Accessories\System Tools All
Users
Administrative Tools All Users:Administrative Tools All Users
MegaRAID Storage Manager All Users:MegaRAID Storage Manager All
Users

Microsoft SQL Server 2005 All Users:Microsoft SQL Server 2005 All Users
Microsoft SQL Server 2005\Analysis Services All Users:Microsoft SQL Server 2005\Analysis Services All Users
Microsoft SQL Server 2005\Configuration Tools All Users:Microsoft SQL Server 2005\Configuration Tools All Users
Microsoft SQL Server 2005\Documentation and Tutorials All Users:Microsoft SQL Server 2005\Documentation and Tutorials All Users
Microsoft SQL Server 2005\Documentation and Tutorials\Tutorials All Users:Microsoft SQL Server 2005\Documentation and Tutorials\Tutorials All Users
Microsoft SQL Server 2005\Performance Tools All Users:Microsoft SQL Server 2005\Performance Tools All Users
Microsoft Visual Studio 2005 All Users:Microsoft Visual Studio 2005 All Users
Microsoft Visual Studio 2005\Visual Studio Tools All Users:Microsoft Visual Studio 2005\Visual Studio Tools All Users
Startup All Users:Startup All Users
Accessories NT AUTHORITY\SYSTEM:Accessories NT AUTHORITY\SYSTEM
Accessories\Accessibility NT AUTHORITY\SYSTEM:Accessories\Accessibility NT AUTHORITY\SYSTEM
Accessories\Entertainment NT AUTHORITY\SYSTEM:Accessories\Entertainment NT AUTHORITY\SYSTEM
Startup NT AUTHORITY\SYSTEM:Startup NT AUTHORITY\SYSTEM
Accessories TPCHA\Administrator:Accessories TPCHA\Administrator
Accessories\Accessibility TPCHA\Administrator:Accessories\Accessibility TPCHA\Administrator
Accessories\Entertainment TPCHA\Administrator:Accessories\Entertainment TPCHA\Administrator
Administrative Tools TPCHA\Administrator:Administrative Tools TPCHA\Administrator
Iometer 2006.07.27 TPCHA\Administrator:Iometer 2006.07.27 TPCHA\Administrator
Startup TPCHA\Administrator:Startup TPCHA\Administrator

[Startup Programs]

Program Command User NameLocation
desktop desktop.ini NT AUTHORITY\SYSTEM Startup
desktop desktop.ini TPCHA\Administrator Startup
desktop desktop.ini .DEFAULT Startup
desktop desktop.ini All Users Common Startup

[OLE Registration]

Object Local Server
Sound (OLE2) sndrec32.exe
Media Clip mplay32.exe
Video Clip mplay32.exe /avi
MIDI Sequence mplay32.exe /mid
Sound Not Available
Media Clip Not Available
WordPad Document "%programfiles%\windows nt\accessories\wordpad.exe"
Bitmap Image mspaint.exe

[Windows Error Reporting]

Time Type Details

[Internet Settings]

[Internet Explorer]

[Following are sub-categories of this main category]
[Summary]

Item Value
Version 6.0.3790.3959
Build 63790.3959
Application Path C:\Program Files\Internet Explorer
Language English (United States)
Active Printer Not Available

Cipher Strength 128-bit
Content Advisor Disabled
IEAK Install No

[File Versions]

File Version Size Date Path Company
actxprxy.dll 6.0.3790.3959 221 KB 2/16/2007 11:02:52 PM C:\WINDOWS\system32 Microsoft Corporation
advpack.dll 6.0.3790.3959 146 KB 2/16/2007 11:03:10 PM C:\WINDOWS\system32 Microsoft Corporation
asctrls.ocx 6.0.3790.1830 147 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
browselc.dll 6.0.3790.1830 63 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
browseui.dll 6.0.3790.3959 1,568 KB 2/16/2007 11:05:24 PM C:\WINDOWS\system32 Microsoft Corporation
cdfview.dll 6.0.3790.3959 216 KB 2/16/2007 11:05:40 PM C:\WINDOWS\system32 Microsoft Corporation
comctl32.dll 5.82.3790.3959 935 KB 2/16/2007 11:09:08 PM C:\WINDOWS\system32 Microsoft Corporation
dxtrans.dll 6.3.3790.3959 325 KB 2/16/2007 11:18:34 PM C:\WINDOWS\system32 Microsoft Corporation
dxtmsft.dll 6.3.3790.3959 549 KB 2/16/2007 11:18:32 PM C:\WINDOWS\system32 Microsoft Corporation
iecont.dll <File Missing> Not Available Not Available Not Available Not Available
iecontlc.dll <File Missing> Not Available Not Available Not Available Not Available
iedkcs32.dll 16.0.3790.3959 417 KB 2/16/2007 11:29:36 PM C:\WINDOWS\system32 Microsoft Corporation
iepeers.dll 6.0.3790.3959 362 KB 2/16/2007 11:29:38 PM C:\WINDOWS\system32 Microsoft Corporation
iesetup.dll 6.0.3790.1830 71 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
ieuinit.inf Not Available 24 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Not Available
iexplore.exe 6.0.3790.1830 94 KB 3/25/2005 7:00:00 AM C:\Program Files\Internet Explorer Microsoft Corporation
imgutil.dll 6.0.3790.3959 61 KB 2/16/2007 11:30:36 PM C:\WINDOWS\system32 Microsoft Corporation
inetcpl.cpl 6.0.3790.3959 431 KB 2/16/2007 11:30:40 PM C:\WINDOWS\system32 Microsoft Corporation
inetcplc.dll 6.0.3790.1830 110 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
inseng.dll 6.0.3790.3959 147 KB 2/16/2007 11:30:52 PM C:\WINDOWS\system32 Microsoft Corporation
mlang.dll 6.0.3790.3959 686 KB 2/16/2007 11:36:42 PM C:\WINDOWS\system32 Microsoft Corporation
msencode.dll <File Missing> Not Available Not Available Not Available Not Available
mshta.exe 6.0.3790.1830 38 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
mshtml.dll 6.0.3790.3959 5,859 KB 2/16/2007 11:38:38 PM C:\WINDOWS\system32 Microsoft Corporation
mshtml.tlb 6.0.3790.1830 1,320 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
mshtmled.dll 6.0.3790.3959 906 KB 2/16/2007 11:38:42 PM C:\WINDOWS\system32 Microsoft Corporation
mshtmler.dll 6.0.3790.1830 56 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
msident.dll 6.0.3790.1830 69 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
msidntld.dll 6.0.3790.1830 16 KB 3/25/2005 7:00:00 AM C:\WINDOWS\system32 Microsoft Corporation
msieftp.dll 6.0.3790.3959 369 KB 2/16/2007 11:38:50 PM C:\WINDOWS\system32 Microsoft Corporation
msrating.dll 6.0.3790.3959 240 KB 2/16/2007 11:39:20 PM C:\WINDOWS\system32 Microsoft Corporation
mstime.dll 6.0.3790.3959 880 KB 2/16/2007 11:39:26 PM C:\WINDOWS\system32 Microsoft Corporation

occache.dll 6.0.3790.3959 126 KB 2/16/2007 11:41:48 PM
C:\WINDOWS\system32 Microsoft Corporation
proctexe.ocx <File Missing> Not Available Not Available
Not Available Not Available
sendmail.dll 6.0.3790.3959 64 KB 2/16/2007 11:54:24 PM
C:\WINDOWS\system32 Microsoft Corporation
shdoclc.dll 6.0.3790.1830 590 KB 3/25/2005 7:00:00 AM
C:\WINDOWS\system32 Microsoft Corporation
shdocvw.dll 6.0.3790.3959 2,381 KB 2/16/2007 11:54:58 PM
C:\WINDOWS\system32 Microsoft Corporation
shfolder.dll 6.0.3790.1830 34 KB 3/25/2005 7:00:00 AM
C:\WINDOWS\system32 Microsoft Corporation
shlwapi.dll 6.0.3790.3959 607 KB 2/16/2007 11:55:32 PM
C:\WINDOWS\system32 Microsoft Corporation
tdc.ocx 1.3.0.3130 91 KB 3/25/2005 7:00:00 AM
C:\WINDOWS\system32 Microsoft Corporation
url.dll 6.0.3790.1830 40 KB 3/25/2005 7:00:00 AM
C:\WINDOWS\system32 Microsoft Corporation
urlmon.dll 6.0.3790.3959 1,063 KB 2/17/2007 12:00:44 AM
C:\WINDOWS\system32 Microsoft Corporation
webcheck.dll 6.0.3790.3959 439 KB 2/17/2007 12:02:26 AM
C:\WINDOWS\system32 Microsoft Corporation
wininet.dll 6.0.3790.3959 1,163 KB 2/17/2007 12:02:54 AM
C:\WINDOWS\system32 Microsoft Corporation

[Connectivity]

Item Value
Connection Preference Never dial

LAN Settings

AutoConfigProxy wininet.dll
AutoProxyDetectMode Disabled
AutoConfigURL
Proxy Disabled
ProxyServer
ProxyOverride

[Cache]

[Following are sub-categories of this main category]
[Summary]

Item Value
Page Refresh Type Automatic
Temporary Internet Files Folder C:\Documents and Settings\Administrator\Local Settings\Temporary Internet Files
Total Disk Space Not Available
Available Disk Space Not Available
Maximum Cache Size Not Available
Available Cache Size Not Available

[List of Objects]

Program File Status CodeBase
No cached object information available

[Content]

[Following are sub-categories of this main category]
[Summary]

Item Value
Content Advisor Disabled

[Personal Certificates]

Issued To Issued By Validity Signature Algorithm
No personal certificate information available

[Other People Certificates]

Issued To Issued By Validity Signature Algorithm
No other people certificate information available

[Publishers]

Name
No publisher information available

[Security]

Zone Security Level
My Computer Custom
Local intranet Custom
Trusted sites Custom
Internet Custom
Restricted sites Custom

Appendix B: Database Build Scripts

BackupDatabase.sql

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

backup database tpch300g
to BackupDev1, BackupDev2, BackupDev3, BackupDev4,
            BackupDev5, BackupDev6, BackupDev7, BackupDev8
with init, maxtransfersize=1048576, BUFFERCOUNT=1000, stats=1
```

BackupDatabase_2.sql

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

backup database tpch300g
to BackupDev9, BackupDev10, BackupDev11, BackupDev12,
            BackupDev13, BackupDev14, BackupDev15, BackupDev16
with init, maxtransfersize=1048576, BUFFERCOUNT=1000, stats=1
```

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
if exists (select name from sysdevices where name = 'BackupDev5')
            exec sp_dropdevice BackupDev5
GO
if exists (select name from sysdevices where name = 'BackupDev6')
            exec sp_dropdevice BackupDev6
GO
if exists (select name from sysdevices where name = 'BackupDev7')
            exec sp_dropdevice BackupDev7
GO
if exists (select name from sysdevices where name = 'BackupDev8')
            exec sp_dropdevice BackupDev8
GO

sp_addumpdevice 'disk', 'BackupDev1', 'E:\tpch300g_backup\TPCH300gbackup1'
GO
sp_addumpdevice 'disk', 'BackupDev2', 'F:\tpch300g_backup\TPCH300gbackup2'
GO
sp_addumpdevice 'disk', 'BackupDev3',
'G:\tpch300g_backup\TPCH300gbackup3'
GO
sp_addumpdevice 'disk', 'BackupDev4',
'H:\tpch300g_backup\TPCH300gbackup4'
GO
sp_addumpdevice 'disk', 'BackupDev5', 'I:\tpch300g_backup\TPCH300gbackup5'
GO
sp_addumpdevice 'disk', 'BackupDev6', 'J:\tpch300g_backup\TPCH300gbackup6'
GO
sp_addumpdevice 'disk', 'BackupDev7',
'O:\tpch300g_backup\TPCH300gbackup7'
GO
sp_addumpdevice 'disk', 'BackupDev8', 'P:\tpch300g_backup\TPCH300gbackup8'
GO
```

CreateBackupDevices_2.sql

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

if exists (select name from sysdevices where name = 'BackupDev9')
            exec sp_dropdevice BackupDev9
GO
if exists (select name from sysdevices where name = 'BackupDev10')
            exec sp_dropdevice BackupDev10
GO
if exists (select name from sysdevices where name = 'BackupDev11')
            exec sp_dropdevice BackupDev11
GO
if exists (select name from sysdevices where name = 'BackupDev12')
            exec sp_dropdevice BackupDev12
GO
if exists (select name from sysdevices where name = 'BackupDev13')
            exec sp_dropdevice BackupDev13
GO
if exists (select name from sysdevices where name = 'BackupDev14')
            exec sp_dropdevice BackupDev14
GO
if exists (select name from sysdevices where name = 'BackupDev15')
            exec sp_dropdevice BackupDev15
GO
if exists (select name from sysdevices where name = 'BackupDev16')
            exec sp_dropdevice BackupDev16
GO

sp_addumpdevice 'disk', 'BackupDev9',
'Q:\tpch300g_backup\TPCH300gbackup9'
GO
sp_addumpdevice 'disk', 'BackupDev10',
'R:\tpch300g_backup\TPCH300gbackup10'
GO
sp_addumpdevice 'disk', 'BackupDev11',
'S:\tpch300g_backup\TPCH300gbackup11'
GO
sp_addumpdevice 'disk', 'BackupDev12',
'T:\tpch300g_backup\TPCH300gbackup12'
GO
sp_addumpdevice 'disk', 'BackupDev13',
'U:\tpch300g_backup\TPCH300gbackup13'
GO
sp_addumpdevice 'disk', 'BackupDev14',
'V:\tpch300g_backup\TPCH300gbackup14'
GO
sp_addumpdevice 'disk', 'BackupDev15',
'W:\tpch300g_backup\TPCH300gbackup15'
GO
sp_addumpdevice 'disk', 'BackupDev16',
'X:\tpch300g_backup\TPCH300gbackup16'
GO
```

CreateDatabase.sql

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

--
--          Drop the existing database
```

```
--

if exists (select name from sysdatabases where name = 'tpch300g')
        drop database tpch300g

CREATE DATABASE tpch300g
ON PRIMARY
(       NAME            = tpch300g_root,
        FILENAME        = 'C:\mp\tpch300g_root.mdf',
        SIZE            = 10MB,
        FILEGROWTH      = 10MB),
FILEGROUP       tpch300g_FG
(       NAME            = tpch300g_1,
        FILENAME        = 'C:\mp\tpch300g1\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_2,
        FILENAME        = 'C:\mp\tpch300g2\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_3,
        FILENAME        = 'C:\mp\tpch300g3\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_4,
        FILENAME        = 'C:\mp\tpch300g4\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_5,
        FILENAME        = 'C:\mp\tpch300g5\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_6,
        FILENAME        = 'C:\mp\tpch300g6\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_7,
        FILENAME        = 'C:\mp\tpch300g7\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_8,
        FILENAME        = 'C:\mp\tpch300g8\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_9,
        FILENAME        = 'C:\mp\tpch300g9\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_10,
        FILENAME        = 'C:\mp\tpch300g10\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_11,
        FILENAME        = 'C:\mp\tpch300g11\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_12,
        FILENAME        = 'C:\mp\tpch300g12\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_13,
        FILENAME        = 'C:\mp\tpch300g13\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_14,
        FILENAME        = 'C:\mp\tpch300g14\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_15,
        FILENAME        = 'C:\mp\tpch300g15\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_16,
        FILENAME        = 'C:\mp\tpch300g16\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_17,
        FILENAME        = 'C:\mp\tpch300g17\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_18,
        FILENAME        = 'C:\mp\tpch300g18\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_19,
        FILENAME        = 'C:\mp\tpch300g19\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_20,
        FILENAME        = 'C:\mp\tpch300g20\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_21,
        FILENAME        = 'C:\mp\tpch300g21\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_22,
        FILENAME        = 'C:\mp\tpch300g22\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = tpch300g_23,
        FILENAME        = 'C:\mp\tpch300g23\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = tpch300g_24,
        FILENAME        = 'C:\mp\tpch300g24\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),


FILEGROUP       LOAD_FG
(       NAME            = loadfg_1,
        FILENAME        = 'C:\mp\loadfg1\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_2,
        FILENAME        = 'C:\mp\loadfg2\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_3,
        FILENAME        = 'C:\mp\loadfg3\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_4,
        FILENAME        = 'C:\mp\loadfg4\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_5,
        FILENAME        = 'C:\mp\loadfg5\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_6,
        FILENAME        = 'C:\mp\loadfg6\',
        SIZE            = 23150MB,
```

```
        FILEGROWTH      = 0),

(       NAME            = loadfg_7,
        FILENAME        = 'C:\mp\loadfg7\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_8,
        FILENAME        = 'C:\mp\loadfg8\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_9,
        FILENAME        = 'C:\mp\loadfg9\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_10,
        FILENAME        = 'C:\mp\loadfg10\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_11,
        FILENAME        = 'C:\mp\loadfg11\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_12,
        FILENAME        = 'C:\mp\loadfg12\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_13,
        FILENAME        = 'C:\mp\loadfg13\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_14,
        FILENAME        = 'C:\mp\loadfg14\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_15,
        FILENAME        = 'C:\mp\loadfg15\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_16,
        FILENAME        = 'C:\mp\loadfg16\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_17,
        FILENAME        = 'C:\mp\loadfg17\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_18,
        FILENAME        = 'C:\mp\loadfg18\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_19,
        FILENAME        = 'C:\mp\loadfg19\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_20,
        FILENAME        = 'C:\mp\loadfg20\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),

(       NAME            = loadfg_21,
        FILENAME        = 'C:\mp\loadfg21\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_22,
        FILENAME        = 'C:\mp\loadfg22\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_23,
        FILENAME        = 'C:\mp\loadfg23\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0),
(       NAME            = loadfg_24,
        FILENAME        = 'C:\mp\loadfg24\',
        SIZE            = 23150MB,
        FILEGROWTH      = 0)

LOG ON
(       NAME            = tpch300glog,
        FILENAME        = 'c:\mp\tpch300glog\',
        SIZE            = 75000MB,
        FILEGROWTH      = 0)
```

CreateFK.sql

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

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

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

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

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

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

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

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

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

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

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 tpch300g_FG

 alter table REGION add constraint PK_R_REGIONKEY primary key
(R_REGIONKEY)
 ON tpch300g_FG

 create index N_REGIONKEY_IDX
       on NATION(N_REGIONKEY)
       with ( fillfactor=100,
            SORT_IN_TEMPDB=ON,
            MAXDOP=16 )
       on tpch300g_FG

 alter table PART add constraint PK_P_PARTKEY primary key (P_PARTKEY)
```

```
 ON tpch300g_FG

 alter table SUPPLIER add constraint PK_S_SUPPKEY primary key
(S_SUPPKEY)
 ON tpch300g_FG

 create index S_NATIONKEY_IDX
            on SUPPLIER(S_NATIONKEY)
            with ( fillfactor=100,
          SORT_IN_TEMPDB=ON,
          MAXDOP=16 )
            on tpch300g_FG

 alter table CUSTOMER add constraint PK_C_CUSTKEY primary key
(C_CUSTKEY)
 ON tpch300g_FG

 alter table PARTSUPP add constraint PK_PS_PARTKEY_PS_SUPPKEY
primary key (PS_PARTKEY, PS_SUPPKEY)
 ON tpch300g_FG

 create clustered index O_ORDERDATE_CLUIDX
            on ORDERS(O_ORDERDATE)
            with ( FILLFACTOR=95,
          SORT_IN_TEMPDB=ON,
          MAXDOP=16 )
            on tpch300g_FG

 alter table ORDERS add constraint PK_O_ORDERKEY primary key
(O_ORDERKEY)
 WITH (FILLFACTOR = 95)
 ON tpch300g_FG
```

CreateIndexes_2.sql

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


 create index PS_SUPPKEY_IDX
      on PARTSUPP(PS_SUPPKEY)
      with ( fillfactor=100,
          SORT_IN_TEMPDB=ON,
          MAXDOP=16 )
      on tpch300g_FG

 create clustered index L_SHIPDATE_CLUIDX
            on LINEITEM(L_SHIPDATE)
            with ( FILLFACTOR=95,
          SORT_IN_TEMPDB=ON,
          MAXDOP=16 )
            on tpch300g_FG

 create index L_ORDERKEY_IDX
            on LINEITEM(L_ORDERKEY)
            with ( FILLFACTOR=95,
          SORT_IN_TEMPDB=ON,
          MAXDOP=16 )
            on tpch300g_FG

 create index L_PARTKEY_IDX
            on LINEITEM(L_PARTKEY)
            with ( FILLFACTOR=95,
          SORT_IN_TEMPDB=ON,
          MAXDOP=16 )
            on tpch300g_FG
```

CreateTables.sql

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

create table PART
          (P_PARTKEY        int                 not null,
           P_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
```

DropLoadFG.sql

```
-- Drop the LOAD_FG

ALTER DATABASE %DBNAME% REMOVE FILE loadfg_1
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_2
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_3
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_4
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_5
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_6
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_7
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_8
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_9
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_10
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_11
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_12
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_13
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_14
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_15
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_16
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_17
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_18
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_19
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_20
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_21
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_22
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_23
ALTER DATABASE %DBNAME% REMOVE FILE loadfg_24
ALTER DATABASE %DBNAME% REMOVE FILEGROUP LOAD_FG
```

RestoreDatabase.sql

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

restore database tpch1g
from BackupDev1, BackupDev2, BackupDev3, BackupDev4,
        BackupDev5, BackupDev6, BackupDev7, BackupDev8
with stats=1, maxtransfersize=1048576, BUFFERCOUNT=1000
```

MoveTempDB.sql

```
alter database tempdb modify file
  (name='tempdev',filename='c:\mp\tempdev1\', size=130MB)
alter database tempdb modify file
  (name='templog',filename='c:\mp\templog1\', size=1024MB)
```

ResizeTempDB.sql

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

alter database tempdb modify file (name=tempdev,size=12700MB,filegrowth=0)

alter database tempdb modify file
(name=templog,size=1152MB,filegrowth=1000)

alter database tempdb add file
(name=tempdev2, filename='c:\mp\tempdev2\',size=12700MB,filegrowth=0),
(name=tempdev3, filename='c:\mp\tempdev3\',size=12700MB,filegrowth=0),
(name=tempdev4, filename='c:\mp\tempdev4\',size=12700MB,filegrowth=0),
(name=tempdev5, filename='c:\mp\tempdev5\',size=12700MB,filegrowth=0),
(name=tempdev6, filename='c:\mp\tempdev6\',size=12700MB,filegrowth=0),
(name=tempdev7, filename='c:\mp\tempdev7\',size=12700MB,filegrowth=0),
(name=tempdev8, filename='c:\mp\tempdev8\',size=12700MB,filegrowth=0),
(name=tempdev9, filename='c:\mp\tempdev9\',size=12700MB,filegrowth=0),
(name=tempdev10, filename='c:\mp\tempdev10\',size=12700MB,filegrowth=0),
(name=tempdev11, filename='c:\mp\tempdev11\',size=12700MB,filegrowth=0),
(name=tempdev12, filename='c:\mp\tempdev12\',size=12700MB,filegrowth=0),
(name=tempdev13, filename='c:\mp\tempdev13\',size=12700MB,filegrowth=0),
(name=tempdev14, filename='c:\mp\tempdev14\',size=12700MB,filegrowth=0),
(name=tempdev15, filename='c:\mp\tempdev15\',size=12700MB,filegrowth=0),
(name=tempdev16, filename='c:\mp\tempdev16\',size=12700MB,filegrowth=0),
(name=tempdev17, filename='c:\mp\tempdev17\',size=12700MB,filegrowth=0),
(name=tempdev18, filename='c:\mp\tempdev18\',size=12700MB,filegrowth=0),
(name=tempdev19, filename='c:\mp\tempdev19\',size=12700MB,filegrowth=0),
(name=tempdev20, filename='c:\mp\tempdev20\',size=12700MB,filegrowth=0),
(name=tempdev21, filename='c:\mp\tempdev21\',size=12700MB,filegrowth=0),
(name=tempdev22, filename='c:\mp\tempdev22\',size=12700MB,filegrowth=0),
(name=tempdev23, filename='c:\mp\tempdev23\',size=12700MB,filegrowth=0),
(name=tempdev24, filename='c:\mp\tempdev24\',size=12700MB,filegrowth=0)
```

Appendix C: Qualification Query Output

Qualification Queries

Query 1

```
L_RETURNFLAG L_LINESTATUS SUM_QTY
SUM_BASE_PRICE          SUM_DISC_PRICE          SUM_CHARGE
AVG_QTY                 AVG_PRICE               AVG_DISC
COUNT_ORDER
------------ ------------ ------------------------ ------------------------ ------------------------
------------------------ ------------------------ ------------------------ ------------------------
--------------------
A            F            151026307.000000         226501015401.550200
215176492146.409180      223786134831.720860      25.503386
38248.586565             0.049997                 5921814
N            F            3931226.000000           5894781841.839997
5599896498.231998        5823867527.830690        25.538387
38294.215975             0.050058                 153934
N            O            297323932.000000         445868009886.018920
423575656799.652100      440525528928.934450      25.500208
38240.201018             0.049997                 11659667
R            F            151132549.000000         226629230916.339450
215301084461.540220      223911875130.769620      25.512904
38257.607769             0.049994                 5923769

(4 row(s) affected)
```

Query 2

```
S_ACCTBAL               S_NAME                  N_NAME
P_PARTKEY   P_MFGR                  S_ADDRESS
S_PHONE         S_COMMENT

------------------------ ------------------------ ------------------------ -----------
------------------------ ---------------------------------------- ---------------
-----------------------------------------------------------------------------------------------------------
--
9990.050000             Supplier#000008890      ROMANIA                 748889
Manufacturer#3          6lmM3OrUukwhKXY0zqypO2qEsgj
29-208-398-4306 ts. unusual deposits haggle furiously along the even

9984.690000             Supplier#000008875      ROMANIA                 576317
Manufacturer#2          hRdOqKqyU,sHq                   29-132-904-4395 ong
the bold pinto beans are furiously blithely slow
9984.690000             Supplier#000008875      ROMANIA                 633748
Manufacturer#5          hRdOqKqyU,sHq                   29-132-904-4395 ong
the bold pinto beans are furiously blithely slow
9968.090000             Supplier#000007191      ROMANIA                 347190
Manufacturer#4          y62XYc3,bw33tqGBua2LqHCd8pIXNL3xO jH,
29-327-376-6773 y bold accounts are quickly among the carefully final idea

9955.050000             Supplier#000008810      UNITED KINGDOM
513707      Manufacturer#5          ,Ot93zDXOFSjWSKsKrT7XJ4YPCP,A
33-527-478-5988 ily according to the carefully express pinto beans. unusual
requests use quickly carefully s
9954.710000             Supplier#000003147      UNITED KINGDOM
763146      Manufacturer#3          jIhW4pvcCF4K41rhP
33-399-127-2845 sleep fluffily after the furi

9938.530000             Supplier#000005359      UNITED KINGDOM
185358      Manufacturer#4          QKuHYh,vZGiwu2FWEJoLDx04
33-429-790-6131 uriously regular requests hag

9937.840000             Supplier#000005969      ROMANIA                 108438
Manufacturer#1          ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa
29-520-692-3537 efully express instructions. regular requests against the slyly fin

9937.840000             Supplier#000005969      ROMANIA                 773391
Manufacturer#5          ANDENSOSmk,miq23Xfb5RWt6dvUcvt6Qa
29-520-692-3537 efully express instructions. regular requests against the slyly fin

9936.220000             Supplier#000005250      UNITED KINGDOM          249
    Manufacturer#4          B3rqp0xbSEim4Mpy2RH J
33-320-228-2957 etect about the furiously final accounts. slyly ironic pinto beans
sleep inside the furiously

... lines deleted

9369.060000             Supplier#000008786      RUSSIA                  398785
Manufacturer#3          MoQKex3SHOANUI9czfzQAbv8Vei
32-460-461-4845 ously regular packages sleep quickly regular dependencies.
carefully thi

(100 row(s) affected)
```

Query 3

```
L_ORDERKEY          REVENUE                 O_ORDERDATE
O_SHIPPRIORITY
-------------------- ------------------------ ------------------------ --------------
23906758            461340.241500           1995-03-14 00:00:00.000 0
11292038            455430.450800           1995-03-07 00:00:00.000 0
9664483             409435.871600           1995-03-10 00:00:00.000 0
2456423             406181.011100           1995-03-05 00:00:00.000 0
3459808             405838.698900           1995-03-04 00:00:00.000 0
11079205            405518.297200           1995-03-10 00:00:00.000 0
20766502            403555.546300           1995-03-11 00:00:00.000 0
21927394            400621.405200           1995-02-06 00:00:00.000 0
10531334            393492.868300           1995-02-07 00:00:00.000 0
12096992            392133.863500           1995-03-12 00:00:00.000 0

(10 row(s) affected)
```

Query 4

```
O_ORDERPRIORITY ORDER_COUNT
--------------- -----------
1-URGENT        42257
2-HIGH          42080
3-MEDIUM        42342
4-NOT SPECIFIED 42281
5-LOW           42107

(5 row(s) affected)
```

Query 5

```
N_NAME                  REVENUE
------------------------ ------------------------
INDONESIA               218627458.003700
CHINA                   217027815.555500
VIETNAM                 210953377.356300
INDIA                   205948019.859900
JAPAN                   194192666.871700

(5 row(s) affected)
```

Query 6

REVENUE

492620021.766600

(1 row(s) affected)

Query 7

SUPP_NATION	CUST_NATION	L_YEAR	REVENUE
FRANCE	GERMANY	1995	211605967.399800
FRANCE	GERMANY	1996	212500572.674300
GERMANY	FRANCE	1995	216935876.450100
GERMANY	FRANCE	1996	214907253.996700

(4 row(s) affected)

Query 8

O_YEAR	MKT_SHARE
1995	0.038724
1996	0.040097

(2 row(s) affected)

Query 9

NATION	O_YEAR	SUM_PROFIT
ALGERIA	1998	124350461.863100
ALGERIA	1997	219878441.358500
ALGERIA	1996	223195863.959900
ALGERIA	1995	216627970.643500
ALGERIA	1994	217074865.314000
ALGERIA	1993	214375601.075700
ALGERIA	1992	212170025.536400
ARGENTINA	1998	114752130.456600
ARGENTINA	1997	202811881.076400
ARGENTINA	1996	204140500.798200
ARGENTINA	1995	201812979.878300

... lines deleted

VIETNAM	1992	203988797.349700

(175 row(s) affected)

Query 10

C_CUSTKEY C_NAME REVENUE C_ACCTBAL
N_NAME C_ADDRESS C_PHONE
C_COMMENT

--- --- --- ---
--- --- ---

104566 Customer#000104566 1196997.959400 4365.140000
EGYPT uYmL1BhBt2Cgi 14-540-497-7494 sly
across the quickly even foxes. daring, final f

125341 Customer#000125341 1103549.349300 4983.510000
GERMANY S29ODD6bceU8QSuuEJznkNaK
17-582-695-5962 arefully even depths. blithely even excuses sleep furiously.
foxes use except the dependencies. ca
112876 Customer#000112876 1094439.494800 -365.170000
SAUDI ARABIA rA13TR18lyKbpKpalnZ
30-768-365-3170 t accounts sleep blithely special instructions. even, regular ideas
haggle. accounts integrate
23560 Customer#000023560 1075607.646900 1439.060000
ETHIOPIA Lp1y9xsfu5Ky5F87Oyl eSoAlez9
15-272-295-2601 above the special asymptotes are even accounts-- even pin

60322 Customer#000060322 1063095.882700 5125.120000
MOROCCO Jv oPdh5ow3HCfk4Ju,h3ZgsQbLlHhIUr4jgXc
25-290-573-1900 pending pinto beans. blithely bold accounts against the spe

26857 Customer#000026857 1045737.622000 7727.260000
ETHIOPIA VUZiEM,n5TDcFwbHLnEIbdNubFkti1Xwpi
15-432-815-1266 s. enticing packages sleep. slyly silent platelets above the
ruthless packages are blithely across
84469 Customer#000084469 1042723.822600 2374.410000
IRAN 6Eu5DjNRMRn5gFuD 20-401-349-8717
final packages. slyly unusual pin

61279 Customer#000061279 1023420.244500 8598.210000
IRAQ 5QKr4zRnIi71Ni92A,cz9Qkc 21-208-585-5676
pinto beans haggle slyly. regular, final requests serve furiously. fluffily ironic
platelets nag
18838 Customer#000018838 1023034.126300 -249.050000
MOZAMBIQUE ic5B2H9otu0YmMPZOH2ejT16Ccf nw8kdIcmks
26-617-516-9195 ons boost. deposits haggle furiously foxes. blithely final
dugouts nag slyly enticingly even packages. carefully re
31546 Customer#000031546 1013925.237600 8876.800000
MOROCCO GiZBtmgmLA bt WSh7Gy33w
25-195-480-6520 posits wake! quickly even theod

103828 Customer#000103828 1011889.194500 2715.080000
BRAZIL VGRsavu2tv8siQ2pWlYSset 12-499-836-1171
fully? furiously dogged accounts believe furiously. express accounts are slyly.
even requests around
15379 Customer#000015379 1010947.377100 8159.500000
IRAQ GB4XN785tYLtiIvIy,bJAKKrfVUQm,2IUA7wPu
21-102-502-2147 ely regular ideas about the express, regular requests print a

60838 Customer#000060838 1009620.544300 2454.770000
BRAZIL 64EaJ5vMAHWJlBOxJklpNc2RJiWE
12-913-494-9813 need to boost against the slyly regular account

8902 Customer#000008902 1006919.585400 9083.390000
INDIA w5kp9vBJEmMZTcDK1D7huYtwERvR
18-206-713-2431 boost slyly slyly ironic foxes. furiously regular ideas sleep
furiously even
85558 Customer#000085558 998615.617600 1153.110000
ROMANIA 0SStabOaTFoy3s4siR ztUwHxd
29-698-267-6025 ing packages hinder carefully. carefully unusual requests grow
against the carefully special in
60202 Customer#000060202 998172.720800 223.030000
UNITED KINGDOM wz WmgkAXQQtBGT,Rw22
33-865-942-1693 he express, unusual packages. slyly final pinto beans haggle
quickly about the slyly silent ideas. carefully
100000 Customer#000100000 995607.080400 4840.820000
PERU Y0wsoJZmr uEKKc9sM 27-488-516-6598
according to the regular pains.

20935 Customer#000020935 990354.523200 3987.670000
ARGENTINA 3v20lkMzcY9ND5IpK4pzgg
11-611-725-3645 pending deposits. blithely silent requests wake according to the
fi

114991 Customer#000114991 983050.132100 7522.540000
ARGENTINA ci0EufuF9wsHGL5TK yHHJowFim
11-954-416-6107 equests boost alongside of the ironic, ironic requests. express, regular deposits use blithely even epitaphs: even
49489 Customer#000049489 982950.441000 1599.100000
CANADA hdoSIa63REEZ4CLrxtcBe1Msi
13-296-149-5774 d, pending theodolites could cajole blithely. quick

(20 row(s) affected)

Query 11

PS_PARTKEY VALUE
--- ---

(0 row(s) affected)

Query 12

L_SHIPMODE HIGH_LINE_COUNT LOW_LINE_COUNT
--- --- ---
MAIL 24858 37191
SHIP 24822 37303

(2 row(s) affected

Query 13

C_COUNT CUSTDIST
--- ---
0 500000
39 3263
38 3149
40 3071
41 3053
37 2998
36 2912
42 2811
43 2703
35 2507
44 2373
77 2366
79 2267
78 2258
34 2202
80 2177
81 2144
82 2113
75 2098
76 2066
83 2012
45 2005
74 1958
84 1880
33 1869
73 1811
46 1792
85 1733
72 1633
32 1618
86 1579
71 1525
47 1477
87 1453
70 1408
31 1322
88 1303
48 1266
69 1218
89 1169
90 1055
68 1039
49 983
30 980
67 958
91 892
50 827
29 775
92 764
66 745
93 647
65 631
28 619
51 604
94 542
64 524
52 499
63 461
95 429
27 401
53 388
96 379
62 363
54 348
97 309
61 280
98 266
26 263
55 237
60 207
99 197
59 193
58 187
25 178
57 173
56 172
100 145
24 126
101 103
102 101
23 73
103 72
104 65
105 47
22 33
107 28
106 28
21 26
109 14
20 13
110 10
111 8
108 8
19 6
113 4
116 2
115 2
114 2
112 2
18 2
125 1
118 1
17 1

(103 row(s) affected)

Query 14

```
PROMO_REVENUE
-----------------------
16.479681

(1 row(s) affected)
```

Query 15

```
S_SUPPKEY   S_NAME                  S_ADDRESS
S_PHONE        TOTAL_REVENUE
----------- ------------------------- ---------------------------------------- ---------------
------------------------
3960        Supplier#000003960        gTlGLliBZ5tejTU
33-532-108-8148 5453663.347400

(1 row(s) affected)
```

Query 16

```
P_BRAND    P_TYPE                    P_SIZE      SUPPLIER_CNT
---------- ------------------------- ----------- ------------
Brand#21   SMALL ANODIZED STEEL      23          59
Brand#12   STANDARD POLISHED COPPER  23          56
Brand#11   STANDARD BRUSHED TIN      23          55
Brand#12   ECONOMY BURNISHED STEEL   3           52
Brand#22   PROMO POLISHED BRASS      45          52
Brand#31   PROMO ANODIZED NICKEL     14          52
Brand#41   LARGE BRUSHED NICKEL      23          52
Brand#44   LARGE PLATED BRASS        19          52
Brand#42   MEDIUM BURNISHED NICKEL   23          51
Brand#12   MEDIUM BRUSHED COPPER     23          48
Brand#21   MEDIUM BURNISHED BRASS    45          48

... lines deleted

Brand#54   STANDARD BURNISHED COPPER 36          3

(27501 row(s) affected)
```

Query 17

```
AVG_YEARLY
-----------------------
1333014.232857

(1 row(s) affected)
```

Query 18

```
C_NAME                  C_CUSTKEY   O_ORDERKEY          O_ORDERDATE
      O_TOTALPRICE
------------------------- ----------- -------------------- -----------------------
------------------------- -----------------------
Customer#000021247        21247       23245603            1994-11-22 00:00:00.000
608162.740000             321.000000
Customer#000128120        128120      4722021             1994-04-07 00:00:00.000
544089.090000             323.000000
Customer#000038918        38918       13185729            1995-08-02 00:00:00.000
542986.090000             305.000000
Customer#000118057        118057      19217698            1993-03-26 00:00:00.000
533490.160000             316.000000
Customer#000041462        41462       8639938             1996-08-30 00:00:00.000
533019.970000             301.000000
Customer#000144617        144617      3043270             1997-02-12 00:00:00.000
530604.440000             317.000000
Customer#000088322        88322       10889601            1992-05-22 00:00:00.000
529513.620000             306.000000
Customer#000012620        12620       23861472            1997-10-30 00:00:00.000
527378.090000             314.000000
Customer#000013940        13940       2232932             1997-04-13 00:00:00.000
522720.610000             304.000000
Customer#000123562        123562      14891232            1992-05-30 00:00:00.000
522087.470000             311.000000

... lines deleted

Customer#000032531        32531       20315522            1998-02-24 00:00:00.000
466697.970000             307.000000

(100 row(s) affected)
```

Query 19

```
REVENUE
-----------------------
12832574.096700

(1 row(s) affected)
```

Query 20

```
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#000000532        ep92hT7VLaVlDKM7lgbj02kIL
Supplier#000000557        jj0wUYh9K3fG5Jhdhrkuy ,4
Supplier#000000688        D fw5ocppmZpYBBIPI718hCihLDZ5KhKX

... lines deleted

Supplier#000009974        7wJ,J5DKcxSU4Kp1cQLpbcAvB5AsvKT

(257 row(s) affected)
```

Query 21

```
S_NAME                  NUMWAIT
------------------------- -----------
Supplier#000005649        61
Supplier#000003806        58
Supplier#000005619        55
Supplier#000002005        54
Supplier#000002344        54
Supplier#000005435        54
Supplier#000002540        53
Supplier#000004489        52
Supplier#000000379        51
Supplier#000001336        51
Supplier#000001452        51
Supplier#000005774        51
Supplier#000007382        51
```

Supplier#000007579	51
Supplier#000009141	51
Supplier#000000415	50
Supplier#000002095	50
Supplier#000005720	50
Supplier#000007131	50
Supplier#000009200	50
Supplier#000000436	49
Supplier#000001152	49
Supplier#000001615	49
Supplier#000002160	49
Supplier#000002301	49
Supplier#000002829	49
Supplier#000003075	49
Supplier#000005896	49
Supplier#000008396	49
Supplier#000000565	48
Supplier#000001046	48
Supplier#000001161	48
Supplier#000002993	48
Supplier#000003372	48
Supplier#000003645	48
Supplier#000003675	48
Supplier#000003992	48
Supplier#000005755	48
Supplier#000000673	47
Supplier#000001251	47
Supplier#000001390	47
Supplier#000003168	47
Supplier#000004109	47
Supplier#000004840	47
Supplier#000006825	47
Supplier#000008134	47
Supplier#000008210	47
Supplier#000008713	47
Supplier#000000188	46
Supplier#000002269	46
Supplier#000002891	46
Supplier#000003220	46
Supplier#000005568	46
Supplier#000005799	46
Supplier#000006210	46
Supplier#000006283	46
Supplier#000006326	46
Supplier#000006629	46
Supplier#000006824	46
Supplier#000007494	46
Supplier#000007497	46
Supplier#000008671	46
Supplier#000009224	46
Supplier#000009435	46
Supplier#000009815	46
Supplier#000000427	45
Supplier#000000486	45
Supplier#000000500	45
Supplier#000000718	45
Supplier#000001007	45
Supplier#000001054	45
Supplier#000001916	45
Supplier#000001925	45
Supplier#000002312	45
Supplier#000002410	45
Supplier#000003101	45
Supplier#000003548	45
Supplier#000003727	45
Supplier#000003949	45
Supplier#000004501	45
Supplier#000005436	45
Supplier#000006027	45
Supplier#000006450	45
Supplier#000006547	45
Supplier#000006667	45
Supplier#000006682	45
Supplier#000006795	45
Supplier#000008490	45
Supplier#000009727	45
Supplier#000009758	45
Supplier#000009835	45
Supplier#000009898	45
Supplier#000000503	44
Supplier#000000788	44
Supplier#000001062	44
Supplier#000001083	44
Supplier#000002596	44
Supplier#000002796	44
Supplier#000005244	44
Supplier#000005583	44

(100 row(s) affected)

Query 22

CNTRYCODE	NUMCUST	TOTACCTBAL
13	8822	66255685.740000
17	9089	68004386.590000
18	9147	68788317.340000
23	9183	68967313.290000
29	9015	67506370.100000
30	9066	68053541.690000
31	9020	67425294.020000

(7 row(s) affected)

Query Substitution Parameters

Power Run Stream 0

1	60						
2	25	BRASS	MIDDLE EAST				
3	FURNITURE	1995-03-31					
4	1997-11-01						
5	ASIA	1996-01-01					
6	1996-01-01	0.08	24				
7	SAUDI ARABIA	INDONESIA					
8	INDONESIA	ASIA	STANDARD PLATED TIN				
9	black						
10	1993-12-01						
11	ETHIOPIA	0.0000003333					
12	REG AIR	FOB	1997-01-01				
13	express	deposits					
14	1997-01-01						
15	1996-01-01						
16	Brand#23	SMALL BRUSHED	15	12	24	40	
43	25	30	6				
17	Brand#32	LG JAR					
18	313						
19	Brand#25	Brand#44	Brand#55	2	15	21	
20	violet	1996-01-01	ROMANIA				
21	MOZAMBIQUE						
22	18	31	16	15	12	28	11

Stream 1

1 68
2 13 NICKEL ASIA
3 MACHINERY 1995-03-17
4 1995-08-01
5 EUROPE 1996-01-01
6 1996-01-01 0.05 24
7 JAPAN ARGENTINA
8 ARGENTINA AMERICA STANDARD
ANODIZED TIN
9 almond
10 1994-09-01
11 CHINA 0.0000003333
12 FOB SHIP 1994-01-01
13 express deposits
14 1997-01-01
15 1994-01-01
16 Brand#53 ECONOMY ANODIZED 18 19 1
17 11 42 35 32
17 Brand#34 LG CAN
18 314
19 Brand#22 Brand#22 Brand#44 7 16 28
20 grey 1994-01-01 INDONESIA
21 INDIA
22 19 29 13 26 24 30 22

Stream 2

1 76
2 50 COPPER AFRICA
3 FURNITURE 1995-03-02
4 1993-05-01
5 MIDDLE EAST 1996-01-01
6 1996-01-01 0.03 25
7 EGYPT CHINA
8 CHINA ASIA PROMO POLISHED TIN
9 tomato
10 1993-06-01
11 FRANCE 0.0000003333
12 MAIL SHIP 1997-01-01
13 express deposits
14 1998-01-01
15 1997-01-01
16 Brand#33 STANDARD PLATED 19 40 12
49 25 32 47 5
17 Brand#31 MED CASE
18 312
19 Brand#34 Brand#15 Brand#44 2 17 25
20 saddle 1993-01-01 UNITED STATES
21 ARGENTENA
22 34 27 12 22 11 19 13

Stream 3

1 84
2 38 STEEL ASIA
3 MACHINERY 1995-03-19
4 1995-11-01
5 AMERICA 1997-01-01
6 1997-01-01 0.08 24
7 VIETNAM IRAN
8 IRAN MIDDLE EAST PROMO BURNISHED NICKEL
9 smoke
10 1994-03-01
11 ROMANIA 0.0000003333
12 TRUCK SHIP 1993-01-01
13 express packages
14 1993-01-01
15 1994-01-01
16 Brand#23 MEDIUM POLISHED 25 11 12
28 20 7 29 31
17 Brand#33 MED JAR
18 313
19 Brand#32 Brand#43 Brand#43 8 18 21
20 cream 1996-01-01 JORDAN
21 CHINA
22 23 25 26 20 30 21 18

Stream 4

1 92
2 26 BRASS AFRICA
3 BUILDING 1995-03-04
4 1993-08-01
5 ASIA 1997-01-01
6 1997-01-01 0.06 24
7 JORDAN BRAZIL
8 BRAZIL AMERICA ECONOMY BRUSHED NICKEL
9 salmon
10 1994-12-01
11 GERMANY 0.0000003333
12 RAIL SHIP 1993-01-01
13 express packages
14 1993-01-01
15 1997-01-01
16 Brand#53 PROMO ANODIZED15 32 21 3
29 47 12 45
17 Brand#35 MED CAN
18 315
19 Brand#34 Brand#31 Brand#32 3 19 28
20 olive 1995-01-01 CANADA
21 IRAN
22 13 18 17 27 12 10 23

Stream 5

1 100
2 14 NICKEL EUROPE
3 MACHINERY 1995-03-21
4 1996-03-01
5 EUROPE 1997-01-01
6 1997-01-01 0.03 25
7 ETHIOPIA ROMANIA
8 ROMANIA EUROPE ECONOMY PLATED NICKEL
9 purple
10 1993-10-01
11 SAUDI ARABIA 0.0000003333
12 AIR SHIP 1993-01-01
13 express packages
14 1993-01-01
15 1995-01-01
16 Brand#33 SMALL BURNISHED 38 15 45
20 25 19 40 39
17 Brand#32 JUMBO BOX
18 312
19 Brand#41 Brand#14 Brand#31 8 20 24
20 beige 1993-01-01 CHINA
21 BRAZIL
22 28 15 21 29 17 18 30

Stream 6

1 108
2 1 TIN AFRICA
3 BUILDING 1995-03-06

4	1993-12-01						
5	MIDDLE EAST		1997-01-01				
6	1997-01-01		0.09	24			
7	RUSSIA	IRAQ					
8	IRAQ	MIDDLE EAST		ECONOMY ANODIZED NICKEL			
9	peach						
10	1994-07-01						
11	INDIA	0.0000003333					
12	SHIP	REG AIR	1994-01-01				
13	special packages						
14	1994-01-01						
15	1997-01-01						
16	Brand#23	LARGE POLISHED		12	24	21	43
8	14	16	48				
17	Brand#34	JUMBO JAR					
18	314						
19	Brand#43	Brand#42	Brand#31	4	10	20	
20	lawn	1996-01-01		INDIA			
21	ROMANIA						
22	12	21	14	30	18	20	13

Appendix D: Refresh Function Source Code

AddEnvVar.vbs

```
Option Explicit
Dim objShell
Dim strVar, strValue, strPathSys, strPathCU
Dim collArgs

strPathSys = "HKLM\SYSTEM\CurrentControlSet\Control\Session
Manager\Environment\"
strPathCU = "HKCU\Environment\"

set collArgs = WScript.Arguments
strVar = collArgs.Item(0)
strValue = collArgs.Item(1)

Set objShell = CreateObject("WScript.Shell")
objShell.RegWrite strPathSys & strVar, strValue, "REG_EXPAND_SZ"
objShell.RegWrite strPathCU & strVar, strValue, "REG_EXPAND_SZ"

WScript.Quit
```

Call_Create_RFStreams

```
Create-RFStreams 64 64 72 144 40 g:\mnt\ntfs 15 Run1 7
```

Clrfs_env

```
TPCH_AUTOMATION_PATH "C:\MSTPCH.205\setup\RF_Procs\CLRFs"
TPCH_AUTOMATION_CMD_PATH "%TPCH_AUTOMATION_PATH%"
TPCH_AUTOMATION_SQL "%TPCH_AUTOMATION_PATH%"
TPCH_AUTOMATION_CONFIG "%TPCH_AUTOMATION_PATH%"
TPCH_AUTOMATION_EXEC "%TPCH_AUTOMATION_PATH%"
TPCH_AUTOMATION_VBS "%TPCH_AUTOMATION_PATH%"
TPCH_MASTERLOG_NAME "MasterLog.txt"
TPCH_OUTPUT_DIR_PREFIX "C:\tpchRuns"
TPCH_EXEC_WINDOW_COLOR "9F"
TPCH_DATABASE "tpch300g"
TPCH_SCALE_FACTOR "300g"
TPCH_SCALE_FACTOR_SIMPLE "300"
TPCH_SQL_SHELL "sqlcmd -Stpcha -E -I -b -w 65535 -l 60 -y 0"
```

Create_RF1_Indices.sql

```
create clustered index NEWORDERS_CLUIDX
        on NEWORDERS (O_ORDERDATE)
        on tpch300g_FG

create clustered index NEWLINEITEM_CLUIDX
        on NEWLINEITEM (L_ORDERKEY)
        on tpch300g_FG
```

Create_RF1_Tables.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 tpch300g_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 tpch300g_FG
```

Create_RF2_Indices.sql

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

Create_RF2_Tables.sql

```
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 tpch300g_FG
set @segment = 1
while @segment <= $(totalSegments)
        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)'
        exec sp_executesql @sql
        set @segment = @segment + 1
        End
```

CreateRF2proc.sql

```
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 @index integer
declare @div integer
declare @mod integer
declare @skip integer
declare @i integer
declare @rangeSum integer
declare @totRangeSize integer
declare @sql nchar(1000)
declare @orderSql nchar(1000)
declare @liSql nchar(1000)

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

set @sql = N'select @sdate = dateadd(day,-1,min(O_ORDERDATE)), @edate =
max(O_ORDERDATE)
from MOD_OLDORDERS'

exec sp_executesql @sql,N'@sdate datetime output, @edate datetime
output',@startdate output, @enddate output

if (@total_executions > @parallel_executions)
	begin
	set @div = (@index-1)/@skip
	set @mod = (@index-1) - @div * @skip
	set @rangeSize = datediff(day, @startdate,
@enddate)/@parallel_executions + 1
	set @totRangeSize = @rangeSize
	set @rangeSum = 0
	set @rangeStart = @div * @rangeSize
	set @i = @mod

	while (@i > 0)
		begin
		set @rangeSize = (@totRangeSize - @rangeSum)/2
		set @rangeSum = @rangeSum + @rangeSize
		set @rangeStart = @rangeStart + @rangeSize
		set @delete_sets = @delete_sets/2
		set @i = @i - 1
		end

	if (@mod + 1 = @skip)
		-- last allocation
		set @rangeSize = @totRangeSize - @rangeSum
	else
		set @rangeSize = (@totRangeSize - @rangeSum)/2

	if (@rangeSize < 0)
		set @rangeSize = 0

	if (@delete_sets <= 0)
		set @delete_sets = 1
	end
else
	begin
	set @rangeSize = datediff(day, @startdate,
@enddate)/@total_executions
	set @rangeStart = @rangeSize * (@index - 1)
	end

set @startdate = dateadd(day, @rangeStart, @startdate)

if (@index < @total_executions)
	set @enddate = dateadd(day, @rangeSize, @startdate)

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

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

--
-- Loop through the order keys only 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)'

SET @orderSql = N'delete from ORDERS where O_ORDERKEY in (select
O_ORDERKEY from MOD_OLDORDERS
WHERE O_ORDERDATE > @startdate AND O_ORDERDATE <= @edate)'

WHILE @startdate < @enddate
	BEGIN
	--
	-- Delete from ORDERS and LINEITEM table
	--
	DELETE_TRANS:
	begin transaction
	exec sp_executesql @liSql, N'@startdate datetime, @edate datetime',
@startdate, @edate
	--
	-- If deadlock victim abort then restart the transaction
	--
	if (@@error = 1205)
		begin
		print 'Delete deadlock - restarting RF2_' +
RTRIM(CONVERT(varchar(30),@current_execution)) + 'transaction'
		rollback transaction
		goto DELETE_TRANS
		end

	exec sp_executesql @orderSql, N'@startdate datetime, @edate
datetime', @startdate, @edate

	--
	-- If deadlock victim abort then restart the transaction
	--
	if (@@error = 1205)
		begin
		print 'Delete deadlock - restarting RF2_' +
RTRIM(CONVERT(varchar(30),@current_execution)) + 'transaction'
		rollback transaction
		goto DELETE_TRANS
		end

	commit transaction

	set @startdate = @edate
	set @edate = dateadd(day, @range, @edate)
	if (@edate > @enddate)
		set @edate = @enddate
	END
```

```
END
GO
```

Create-RF1_Whole

```
setlocal
set TPCH_NUM_LOAD_STREAMS=%1
set TPCH_NUM_LOAD_FILES=%2
set TPCH_NUM_EXEC_STREAMS=%3
set TPCH_NUM_EXEC_TOTAL=%4
set TPCH_EXEC_BATCH_SIZE=%5
set TPCH_FF_LOCN=%6
set TPCH_RF_STREAM_NUM=%7
set /a TPCH_UPDATESET=%TPCH_RF_STREAM_NUM% + 1
set TPCH_RUN_NUM=%8
set TPCH_COMMENT_STR=%9
set TPCH_FF_LOCN=%TPCH_FF_LOCN%
set

TPCH_OUTPUT_DIR=%TPCH_OUTPUT_DIR_PREFIX%\%TPCH_RUN_NU
M%
set

TPCH_MASTERLOG_FILE=%TPCH_OUTPUT_DIR%\%TPCH_MASTERLO
G_NAME%

if not exist %TPCH_OUTPUT_DIR% md %TPCH_OUTPUT_DIR%
pushd %TPCH_OUTPUT_DIR%

if not exist Create_RF1_Tables.sql copy
%TPCH_AUTOMATION_SQL%\Create_RF1_Tables.sql
if not exist Create_RF1_Indices.sql copy
%TPCH_AUTOMATION_SQL%\Create_RF1_Indices.sql
set /A LOCAL_CURR_STREAM = 1

:outerLoopLoadCreate
set /A LOCAL_CURR_LOAD = %LOCAL_CURR_STREAM%
if exist

(RF1_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt) del

RF1_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt

:innerLoopLoadCreate
echo :setvar ffLoc %TPCH_FF_LOCN% >

load_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_LOAD%.sq
l
echo :setvar partNo %LOCAL_CURR_LOAD% >>

load_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_LOAD%.sq
l
type %TPCH_AUTOMATION_SQL%\Load_RF1_tmpl.sql >>

load_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_LOAD%.sq
l
echo

load_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_LOAD%.sq
l >>

RF1_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt

set /A LOCAL_CURR_LOAD = %LOCAL_CURR_LOAD% +
%TPCH_NUM_LOAD_STREAMS%
if %LOCAL_CURR_LOAD% LEQ %TPCH_NUM_LOAD_FILES% goto
innerLoopLoadCreate

set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% + 1
if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_LOAD_STREAMS%
goto outerLoopLoadCreate
set /A LOCAL_CURR_STREAM = 1

:outerLoopExecCreate
set /A LOCAL_CURR_EXEC = %LOCAL_CURR_STREAM%
if exist

(RF1_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt) del

RF1_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt

:innerLoopExecCreate
echo :setvar currExec %LOCAL_CURR_EXEC% >

exec_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
echo :setvar batchSize %TPCH_EXEC_BATCH_SIZE% >>

exec_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
echo :setvar numParallelExecs %TPCH_NUM_EXEC_STREAMS% >>

exec_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
echo :setvar totalExecs %TPCH_NUM_EXEC_TOTAL% >>

exec_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
type %TPCH_AUTOMATION_SQL%\Execute_RF1_tmpl.sql>>

exec_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
echo

exec_RF1_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l >>

RF1_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt
set /A LOCAL_CURR_EXEC = %LOCAL_CURR_EXEC% +
%TPCH_NUM_EXEC_STREAMS%

if %LOCAL_CURR_EXEC% LEQ %TPCH_NUM_EXEC_TOTAL% goto
innerLoopExecCreate
set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% + 1
if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_EXEC_STREAMS%
goto outerLoopExecCreate
endlocal
```

Create-RF2_Whole

```
setlocal
set TPCH_NUM_LOAD_STREAMS=%1
set TPCH_NUM_LOAD_FILES=%2
set TPCH_NUM_EXEC_STREAMS=%3
set TPCH_NUM_EXEC_TOTAL=%4
set TPCH_EXEC_BATCH_SIZE=%5
set TPCH_FF_LOCN=%6
set TPCH_RF_STREAM_NUM=%7
set /a TPCH_UPDATESET=%7+1
set TPCH_RUN_NUM=%8
set TPCH_COMMENT_STR=%9
set

TPCH_OUTPUT_DIR=%TPCH_OUTPUT_DIR_PREFIX%\%TPCH_RUN_NU
M%
set
```

```
TPCH_MASTERLOG_FILE=%TPCH_OUTPUT_DIR%\%TPCH_MASTERLO
G_NAME%


if not exist %TPCH_OUTPUT_DIR% md %TPCH_OUTPUT_DIR%
pushd %TPCH_OUTPUT_DIR%

echo :setvar totalSegments %TPCH_NUM_LOAD_FILES% >
Create_RF2_Tables.sql
type %TPCH_AUTOMATION_SQL%\Create_RF2_Tables.sql >>
Create_RF2_Tables.sql
if not exist Increment_Update_Set.sql copy
%TPCH_AUTOMATION_SQL%\Increment_Update_Set.sql
if not exist Create_RF2_Indices.sql copy
%TPCH_AUTOMATION_SQL%\Create_RF2_Indices.sql
set /A LOCAL_CURR_STREAM = 1

:outerLoopLoadCreate
set /A LOCAL_CURR_LOAD = %LOCAL_CURR_STREAM%
if exist

(RF2_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt) del

RF2_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt

:innerLoopLoadCreate
echo :setvar ffLoc %TPCH_FF_LOCN% >

load_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_LOAD%.sq
l
echo :setvar partNo %LOCAL_CURR_LOAD% >>

load_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_LOAD%.sq
l
type %TPCH_AUTOMATION_SQL%\Load_RF2_tmpl.sql >>

load_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_LOAD%.sq
l
echo

load_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_LOAD%.sq
l >>

RF2_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt

set /A LOCAL_CURR_LOAD = %LOCAL_CURR_LOAD% +
%TPCH_NUM_LOAD_STREAMS%
if %LOCAL_CURR_LOAD% LEQ %TPCH_NUM_LOAD_FILES% goto
innerLoopLoadCreate
set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% + 1
if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_LOAD_STREAMS%
goto outerLoopLoadCreate
set /A LOCAL_CURR_STREAM = 1

:outerLoopExecCreate
set /A LOCAL_CURR_EXEC = %LOCAL_CURR_STREAM%
if exist

(RF2_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt) del

RF2_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt

:innerLoopExecCreate
echo :setvar currExec %LOCAL_CURR_EXEC% >

exec_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
echo :setvar batchSize %TPCH_EXEC_BATCH_SIZE% >>

exec_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
echo :setvar numParallelExecs %TPCH_NUM_EXEC_STREAMS% >>

exec_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
echo :setvar totalExecs %TPCH_NUM_EXEC_TOTAL% >>

exec_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
type %TPCH_AUTOMATION_SQL%\Execute_RF2_tmpl.sql>>

exec_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l
echo

exec_RF2_input.%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_EXEC%.sq
l >>

RF2_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt

set /A LOCAL_CURR_EXEC = %LOCAL_CURR_EXEC% +
%TPCH_NUM_EXEC_STREAMS%
if %LOCAL_CURR_EXEC% LEQ %TPCH_NUM_EXEC_TOTAL% goto
innerLoopExecCreate
set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% + 1
if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_EXEC_STREAMS%
goto outerLoopExecCreate
Endlocal
```

Create-RF_Streams

```
setlocal
set TPCH_NUM_LOAD_STREAMS=%1
set TPCH_NUM_LOAD_FILES=%2
set TPCH_NUM_EXEC_STREAMS=%3
set TPCH_NUM_EXEC_TOTAL=%4
set TPCH_EXEC_BATCH_SIZE=%5
set TPCH_FF_LOCN=%6
set TPCH_NUM_RF_STREAMS=%7
set TPCH_RUN_NUM=%8
set TPCH_SWITCH_TO_RUN2=%9
set

TPCH_OUTPUT_DIR=%TPCH_OUTPUT_DIR_PREFIX%\%TPCH_RUN_NU
M%
set

TPCH_MASTERLOG_FILE=%TPCH_OUTPUT_DIR%\%TPCH_MASTERLO
G_NAME%

if not exist %TPCH_OUTPUT_DIR% md %TPCH_OUTPUT_DIR%
pushd %TPCH_OUTPUT_DIR%
set /A LOCAL_CURR_STREAM = 0

:outerRFLoop
if %LOCAL_CURR_STREAM% GTR %TPCH_SWITCH_TO_RUN2% set
TPCH_RUN_NUM=Run2
call %TPCH_AUTOMATION_PATH%\Create-RF1_Whole.cmd
%TPCH_NUM_LOAD_STREAMS% %TPCH_NUM_LOAD_FILES%
%TPCH_NUM_EXEC_STREAMS% %TPCH_NUM_EXEC_TOTAL%
%TPCH_EXEC_BATCH_SIZE% %TPCH_FF_LOCN%
%LOCAL_CURR_STREAM% %TPCH_RUN_NUM%
Throughput_RF1_Stream%LOCAL_CURR_STREAM%
call %TPCH_AUTOMATION_PATH%\Create-RF2_Whole.cmd
%TPCH_NUM_LOAD_STREAMS% %TPCH_NUM_LOAD_FILES%
%TPCH_NUM_EXEC_STREAMS% %TPCH_NUM_EXEC_TOTAL%
%TPCH_EXEC_BATCH_SIZE% %TPCH_FF_LOCN%
%LOCAL_CURR_STREAM% %TPCH_RUN_NUM%
Throughput_RF2_Stream%LOCAL_CURR_STREAM%

set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% + 1
```

```
if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_RF_STREAMS% goto
outerRFLoop
endlocal
```

ExecSQLFileList

```
@echo off

setlocal ENABLEDELAYEDEXPANSION


set TPCH_INPUT_LIST=%1
set TPCH_INPUT_LIST_DIR=%~dp1
set TPCH_INPUT_LIST_NAME=%~n1
set TPCH_RUN_NUM=%2
set

TPCH_OUTPUT_DIR=%TPCH_OUTPUT_DIR_PREFIX%\%TPCH_RUN_NU
M%
set

TPCH_MASTERLOG_FILE=%TPCH_OUTPUT_DIR%\%TPCH_MASTERLO
G_NAME%
set TPCH_COLLECT_SHOWPLAN=%3
set TPCH_COLLECT_PERFMON=%4
set TPCH_WRITE_TO_MASTER=%5
if not exist %TPCH_OUTPUT_DIR% md %TPCH_OUTPUT_DIR%
COLOR %TPCH_EXEC_WINDOW_COLOR%

for /F "tokens=1" %%i in (%TPCH_INPUT_LIST%) do (
	set TPCH_CURRENT_FILE_NAME=%%~nxi
	set TPCH_CURRENT_FILE=%TPCH_INPUT_LIST_DIR%\%%~nxi
	set
TPCH_CURRENT_OUTPUT_FILE=%TPCH_OUTPUT_DIR%\%%~ni.out
	set TPCH_EXEC_FILE=!TPCH_CURRENT_FILE!

	if not exist !TPCH_CURRENT_FILE! (
		echo %CD%
		echo File !TPCH_CURRENT_FILE! missing
		pause
		goto :missingfile
	)
	title %%~ni
	if /i '%TPCH_WRITE_TO_MASTER%'=='yes' (
		echo !DATE! !TIME! Start: Executing
!TPCH_CURRENT_FILE! >> %TPCH_MASTERLOG_FILE%
		)
	set ERRORLEVEL=0
	%TPCH_SQL_SHELL% -d %TPCH_DATABASE% -i
!TPCH_EXEC_FILE! -o !TPCH_CURRENT_OUTPUT_FILE!
	echo %ERRORLEVEL%
	if not errorlevel 0 goto :errorexit
	if /i '%TPCH_WRITE_TO_MASTER%'=='yes' (
		echo !DATE! !TIME! End : Executing
!TPCH_CURRENT_FILE! >> %TPCH_MASTERLOG_FILE%
		)
	)

goto finish

:errorexit
echo Program returned with error when executing %TPCH_CURRENT_FILE%
>> %TPCH_MASTERLOG_FILE%
goto finish
:missingfile
echo File !TPCH_CURRENT_FILE! missing >>
%TPCH_MASTERLOG_FILE%
goto finish
:finish
Endlocal
```

ExecSQLFileListSemaphore

```
@echo off

setlocal
set TPCH_SEMAPHORE_NAME=%1
shift
call %TPCH_AUTOMATION_CMD_PATH%\execSQLFileList %1 %2 %3 %4
%5 %6 %7 %8 %9
%TPCH_AUTOMATION_EXEC%\semaphore -release
%TPCH_SEMAPHORE_NAME%
exit
endlocal
```

Execute_RF1_tmp.sql

```
EXEC RF1 $(currExec), $(batchSize), $(numParallelExecs), $(totalExecs)
```

Execute_RF2_tmpl.sql

```
EXEC RF2 $(currExec), $(batchSize), $(numParallelExecs), $(totalExecs)
```

Increment_Update_Set.sql

```
UPDATE TPCH_AUX_TABLE SET updateset=updateset+1
go
select * from TPCH_AUX_TABLE
go
```

installEnvVariables

```
Rem This file installs the environment variables in
%TPCH_ENV_INPUT_LIST%

Setlocal

echo off

set TPCH_ENV_INPUT_LIST=%1

for /F "tokens=1,* delims= " %%i in (%TPCH_ENV_INPUT_LIST%) do (

	set TPCH_CURRENT_ENV_VAR=%%i
	set TPCH_CURRENT_ENV_VALUE=%%j

	REM echo !TPCH_START_LINE!

	cscript addEnvVar.vbs !TPCH_CURRENT_ENV_VAR!
!TPCH_CURRENT_ENV_VALUE!
	REM reg add "HKLM\SYSTEM\CurrentControlSet\Control\Session
Manager\Environment" /v !TPCH_CURRENT_ENV_VAR! /t
REG_EXPAND_SZ /d !TPCH_CURRENT_ENV_VALUE! /f /s
	REM reg add "HKCU\Environment" /v
!TPCH_CURRENT_ENV_VAR! /t  REG_EXPAND_SZ /d
!TPCH_CURRENT_ENV_VALUE! /f

	REM setx !TPCH_CURRENT_ENV_VAR!
!TPCH_CURRENT_ENV_VALUE! /M
	REM setx !TPCH_CURRENT_ENV_VAR!
!TPCH_CURRENT_ENV_VALUE!

	REM echo %ERRORLEVEL%
	if not errorlevel 0 goto :errorexit

)

goto finish

:errorexit
goto finish

:finish
```

```
endlocal
```

Load_RF1_tmpl.sql

```
declare @sql nchar(1000)
declare @set integer

select @set = updateset from TPCH_AUX_TABLE
set @sql = 'bulk insert NEWLINEITEM from "$(ffLoc)\Lineitem.tbl.u' +
RTRIM(CONVERT(varchar(30),@set))+ '.$(partNo)" with
(FieldTerminator = "|", RowTerminator ="\n",tablock)'
exec sp_executesql @sql
set @sql = 'bulk insert NEWORDERS from "$(ffLoc)\Orders.tbl.u' +
RTRIM(CONVERT(varchar(30), @set)) + '.$(partNo)" with
(FieldTerminator = "|", RowTerminator ="\n",tablock)'
exec sp_executesql @sql
```

Load_RF2_tmpl.sql

```
declare @set integer
declare @sql nchar(1000)
select @set = updateset from TPCH_AUX_TABLE
set @sql = 'bulk insert OLDORDERS_$(partNo) from "$(ffLoc)\Delete.u' +
RTRIM(CONVERT(varchar(30),@set)) + '.$(partNo)" with
(RowTerminator ="|",tablock)'

exec sp_executesql @sql
insert into MOD_OLDORDERS (O_ORDERDATE, O_ORDERKEY) (select
B.O_ORDERDATE, B.O_ORDERKEY from
            OLDORDERS_$(partNo) A, ORDERS B where A.O_ORDERKEY =
B.O_ORDERKEY)
```

precreatefiles_300G

```
call Create-RF1_Whole.cmd 32 64 32 128 40 N:\RF_Flat_File 0 rf1_run1
Run1_power_rf1
call Create-RF1_Whole.cmd 32 64 32 128 40 N:\RF_Flat_File 7 rf1_run2
Run2_power_rf1
call Create-RF1_Whole.cmd 32 64 32 128 40 N:\RF_Flat_File 14 rf1_run3
Run3_power_rf1
call Create-RF2_Whole.cmd 32 64 32 128 40 N:\RF_Flat_File 0 rf2_run1
Run1_power_rf2
call Create-RF2_Whole.cmd 32 64 32 128 40 N:\RF_Flat_File 7 rf2_run2
Run2_power_rf2
call Create-RF2_Whole.cmd 32 64 32 128 40 N:\RF_Flat_File 14 rf2_run3
Run3_power_rf2
call Create-RF_Streams.cmd 32 64 32 128 40 N:\RF_Flat_File 6 rfstreams_run1
RFstream1
call Create-RF_Streams.cmd 32 64 32 128 40 N:\RF_Flat_File 14 rfstreams_run2
RFstream2
```

RF1 Command Invocations

```
C:\MSTPCH.205\setup\RF_Procs\CLRFs\RF1_Whole.cmd 32 64 32 128 40
N:\RF_Flat_File 0 rf1_run1 Run1_Power_rf1
C:\MSTPCH.205\setup\RF_Procs\CLRFs\RF1_Whole 32 64 32 128 40
N:\RF_Flat_File 7 rf1_run2 Run2_Power_rf1
C:\MSTPCH.205\setup\RF_Procs\CLRFs\RF1_Whole 32 64 32 128 40
N:\RF_Flat_File 14 rf1_run3 Run3_Power_rf1
C:\MSTPCH.205\setup\RF_Procs\cLRFs\RF2_Whole 32 64 32 128 40
N:\RF_Flat_File 0 rf2_run1 Run1_Power_rf2
C:\MSTPCH.205\setup\RF_Procs\cLRFs\RF2_Whole 32 64 32 128 40
N:\RF_Flat_File 7 rf2_run2 Run2_Power_rf2
C:\MSTPCH.205\setup\RF_Procs\cLRFs\RF2_Whole 32 64 32 128 40
N:\RF_Flat_File 14 rf2_run3 Run3_Power_rf2
C:\MSTPCH.205\setup\RF_Procs\cLRFs\RFStreams_Run1 32 64 32 128 40
N:\RF_Flat_File 6 RFstreams_Run1 RFstream1
C:\MSTPCH.205\setup\RF_Procs\cLRFs\RFStreams_Run2 32 64 32 128 40
N:\RF_Flat_File 13 RFstreams_run2 RFstream2
```

RF1_Whole

```
setlocal

echo off

set TPCH_NUM_LOAD_STREAMS=%1
set TPCH_NUM_LOAD_FILES=%2
set TPCH_NUM_EXEC_STREAMS=%3
set TPCH_NUM_EXEC_TOTAL=%4
set TPCH_EXEC_BATCH_SIZE=%5
set TPCH_FF_LOCN=%6
set TPCH_RF_STREAM_NUM=%7
set TPCH_RUN_NUM=%8
set TPCH_COMMENT_STR=%9
set

TPCH_OUTPUT_DIR=%TPCH_OUTPUT_DIR_PREFIX%\%TPCH_RUN_NU
M%
set

TPCH_MASTERLOG_FILE=%TPCH_OUTPUT_DIR%\%TPCH_MASTERLO
G_NAME%

if not exist %TPCH_OUTPUT_DIR% md %TPCH_OUTPUT_DIR%
pushd %TPCH_OUTPUT_DIR%

echo %DATE% %TIME% Start: Executing RF1 %TPCH_COMMENT_STR%
>> %TPCH_MASTERLOG_FILE%
%TPCH_SQL_SHELL% -d %TPCH_DATABASE% -i create_RF1_tables.sql >>
create_RF1_tables.out

set

TPCH_SEMAPHORE_NAME=Load.%TPCH_RF_STREAM_NUM%.%TPCH_
RUN_NUM%

start %TPCH_AUTOMATION_CMD_PATH%\WaitForSemaphore.cmd
%TPCH_SEMAPHORE_NAME% %TPCH_NUM_LOAD_STREAMS%
%TPCH_SEMAPHORE_NAME%_all
set /A LOCAL_CURR_STREAM = 1

:outerLoopLoadRun
echo        start
%TPCH_AUTOMATION_CMD_PATH%\execSQLFileListSemaphore.cmd
%TPCH_SEMAPHORE_NAME%

RF1_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt %TPCH_RUN_NUM% no no no >> c:\ListCommands.txt
            start
%TPCH_AUTOMATION_CMD_PATH%\execSQLFileListSemaphore.cmd
%TPCH_SEMAPHORE_NAME%

RF1_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt %TPCH_RUN_NUM% no no no
            set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM%
+1

if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_LOAD_STREAMS%
goto outerLoopLoadRun

%TPCH_AUTOMATION_EXEC%\semaphore -wait
%TPCH_SEMAPHORE_NAME%_all -count 1

%TPCH_SQL_SHELL% -d %TPCH_DATABASE% -i create_RF1_indices.sql
>> create_indices_RF1.out

set

TPCH_SEMAPHORE_NAME=Exec.%TPCH_RF_STREAM_NUM%.%TPCH_
RUN_NUM%
```

```
start %TPCH_AUTOMATION_CMD_PATH%\WaitForSemaphore.cmd
%TPCH_SEMAPHORE_NAME% %TPCH_NUM_EXEC_STREAMS%
%TPCH_SEMAPHORE_NAME%_all
set /A LOCAL_CURR_STREAM = 1

:outerLoopExecRun

echo	start
%TPCH_AUTOMATION_CMD_PATH%\execSQLFileListSemaphore.cmd
%TPCH_SEMAPHORE_NAME%

RF1_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt %TPCH_RUN_NUM% no no no >> c:\ListCommands.txt
	start
%TPCH_AUTOMATION_CMD_PATH%\execSQLFileListSemaphore.cmd
%TPCH_SEMAPHORE_NAME%

RF1_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt %TPCH_RUN_NUM% no no no
	set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% +
1

if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_EXEC_STREAMS%
goto outerLoopExecRun

%TPCH_AUTOMATION_EXEC%\semaphore -wait
%TPCH_SEMAPHORE_NAME%_all -count 1

echo %DATE% %TIME% End : Executing RF1 %TPCH_COMMENT_STR%
>> %TPCH_MASTERLOG_FILE%
endlocal
```

RF2_Whole

```
setlocal

echo off
set TPCH_NUM_LOAD_STREAMS=%1
set TPCH_NUM_LOAD_FILES=%2
set TPCH_NUM_EXEC_STREAMS=%3
set TPCH_NUM_EXEC_TOTAL=%4
set TPCH_EXEC_BATCH_SIZE=%5
set TPCH_FF_LOCN=%6
set TPCH_RF_STREAM_NUM=%7
set TPCH_RUN_NUM=%8
set TPCH_COMMENT_STR=%9
set

TPCH_OUTPUT_DIR=%TPCH_OUTPUT_DIR_PREFIX%\%TPCH_RUN_NU
M%
set

TPCH_MASTERLOG_FILE=%TPCH_OUTPUT_DIR%\%TPCH_MASTERLO
G_NAME%

if not exist %TPCH_OUTPUT_DIR% md %TPCH_OUTPUT_DIR%
pushd %TPCH_OUTPUT_DIR%

echo %DATE% %TIME% Start: Executing RF2 %TPCH_COMMENT_STR%
>> %TPCH_MASTERLOG_FILE%
%TPCH_SQL_SHELL% -d %TPCH_DATABASE% -icreate_RF2_tables.sql >>
create_RF2_tables.out

set

TPCH_SEMAPHORE_NAME=Load.%TPCH_RF_STREAM_NUM%.%TPCH_
RUN_NUM%

start %TPCH_AUTOMATION_CMD_PATH%\WaitForSemaphore.cmd
%TPCH_SEMAPHORE_NAME% %TPCH_NUM_LOAD_STREAMS%
%TPCH_SEMAPHORE_NAME%_all
set /A LOCAL_CURR_STREAM = 1

:outerLoopLoadRun
	start
%TPCH_AUTOMATION_CMD_PATH%\execSQLFileListSemaphore.cmd
%TPCH_SEMAPHORE_NAME%

RF2_Load_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREAM
%.txt %TPCH_RUN_NUM% no no no
	set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% +
1

if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_LOAD_STREAMS%
goto outerLoopLoadRun

%TPCH_AUTOMATION_EXEC%\semaphore -wait
%TPCH_SEMAPHORE_NAME%_all -count 1

%TPCH_SQL_SHELL% -d %TPCH_DATABASE% -i create_RF2_indices.sql
>> create_indices_RF2.out

set

TPCH_SEMAPHORE_NAME=Exec.%TPCH_RF_STREAM_NUM%.%TPCH_
RUN_NUM%

start %TPCH_AUTOMATION_CMD_PATH%\WaitForSemaphore.cmd
%TPCH_SEMAPHORE_NAME% %TPCH_NUM_EXEC_STREAMS%
%TPCH_SEMAPHORE_NAME%_all
set /A LOCAL_CURR_STREAM = 1

:outerLoopExecRun

	start
%TPCH_AUTOMATION_CMD_PATH%\execSQLFileListSemaphore.cmd
%TPCH_SEMAPHORE_NAME%

RF2_Execute_List_%TPCH_RF_STREAM_NUM%.%LOCAL_CURR_STREA
M%.txt %TPCH_RUN_NUM% no no no
	set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% +
1

if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_EXEC_STREAMS%
goto outerLoopExecRun

%TPCH_AUTOMATION_EXEC%\semaphore -wait
%TPCH_SEMAPHORE_NAME%_all -count 1

echo %DATE% %TIME% End : Executing RF2 %TPCH_COMMENT_STR%
>> %TPCH_MASTERLOG_FILE%
%TPCH_SQL_SHELL% -d %TPCH_DATABASE%
-i%TPCH_AUTOMATION_SQL%\Increment_Update_Set.sql >>
Increment_Update_Set.out
Endlocal
```

RFStreams_Run1

```
@echo off
setlocal
set TPCH_NUM_LOAD_STREAMS=%1
set TPCH_NUM_LOAD_FILES=%2
set TPCH_NUM_EXEC_STREAMS=%3
set TPCH_NUM_EXEC_TOTAL=%4
set TPCH_EXEC_BATCH_SIZE=%5
set TPCH_FF_LOCN=%6
set TPCH_NUM_RF_STREAMS=%7
set TPCH_RUN_NUM=%8
set

TPCH_OUTPUT_DIR=%TPCH_OUTPUT_DIR_PREFIX%\%TPCH_RUN_NU
M%
set

TPCH_MASTERLOG_FILE=%TPCH_OUTPUT_DIR%\%TPCH_MASTERLO
G_NAME%
```

```
if not exist %TPCH_OUTPUT_DIR% md %TPCH_OUTPUT_DIR%
pushd %TPCH_OUTPUT_DIR%
set /A LOCAL_CURR_STREAM = 1

:outerRFLoop
call %TPCH_AUTOMATION_CMD_PATH%\RF1_Whole.cmd
%TPCH_NUM_LOAD_STREAMS% %TPCH_NUM_LOAD_FILES%
%TPCH_NUM_EXEC_STREAMS% %TPCH_NUM_EXEC_TOTAL%
%TPCH_EXEC_BATCH_SIZE% %TPCH_FF_LOCN%
%LOCAL_CURR_STREAM% %TPCH_RUN_NUM%
Throughput_RF1_Stream%LOCAL_CURR_STREAM%
call %TPCH_AUTOMATION_CMD_PATH%\RF2_Whole.cmd
%TPCH_NUM_LOAD_STREAMS% %TPCH_NUM_LOAD_FILES%
%TPCH_NUM_EXEC_STREAMS% %TPCH_NUM_EXEC_TOTAL%
%TPCH_EXEC_BATCH_SIZE% %TPCH_FF_LOCN%
%LOCAL_CURR_STREAM% %TPCH_RUN_NUM%
Throughput_RF2_Stream%LOCAL_CURR_STREAM%
set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% + 1
if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_RF_STREAMS% goto
outerRFLoop
Endlocal
```

RFStreams_Run2

```
@echo off
setlocal
set TPCH_NUM_LOAD_STREAMS=%1
set TPCH_NUM_LOAD_FILES=%2
set TPCH_NUM_EXEC_STREAMS=%3
set TPCH_NUM_EXEC_TOTAL=%4
set TPCH_EXEC_BATCH_SIZE=%5
set TPCH_FF_LOCN=%6
set TPCH_NUM_RF_STREAMS=%7
set TPCH_RUN_NUM=%8
set

TPCH_OUTPUT_DIR=%TPCH_OUTPUT_DIR_PREFIX%\%TPCH_RUN_NU
M%
set

TPCH_MASTERLOG_FILE=%TPCH_OUTPUT_DIR%\%TPCH_MASTERLO
G_NAME%

if not exist %TPCH_OUTPUT_DIR% md %TPCH_OUTPUT_DIR%
pushd %TPCH_OUTPUT_DIR%
set /A LOCAL_CURR_STREAM = 8

:outerRFLoop
call %TPCH_AUTOMATION_CMD_PATH%\RF1_Whole.cmd
%TPCH_NUM_LOAD_STREAMS% %TPCH_NUM_LOAD_FILES%
%TPCH_NUM_EXEC_STREAMS% %TPCH_NUM_EXEC_TOTAL%
%TPCH_EXEC_BATCH_SIZE% %TPCH_FF_LOCN%
%LOCAL_CURR_STREAM% %TPCH_RUN_NUM%
Throughput_RF1_Stream%LOCAL_CURR_STREAM%
call %TPCH_AUTOMATION_CMD_PATH%\RF2_Whole.cmd
%TPCH_NUM_LOAD_STREAMS% %TPCH_NUM_LOAD_FILES%
%TPCH_NUM_EXEC_STREAMS% %TPCH_NUM_EXEC_TOTAL%
%TPCH_EXEC_BATCH_SIZE% %TPCH_FF_LOCN%
%LOCAL_CURR_STREAM% %TPCH_RUN_NUM%
Throughput_RF2_Stream%LOCAL_CURR_STREAM%
set /A LOCAL_CURR_STREAM = %LOCAL_CURR_STREAM% + 1
if %LOCAL_CURR_STREAM% LEQ %TPCH_NUM_RF_STREAMS% goto
outerRFLoop
Endlocal
```

Run_rf1whole

```
rf1_whole 32 64 32 128 40 N:\RF_Flat_File 6 Run1_1 vinay3
```

WaitForSemaphore

```
setlocal
set TPCH_Group=%1
set waitcount=%2
set TPCH_Sentinel=%3
%TPCH_AUTOMATION_EXEC%\semaphore -wait %TPCH_Group% -count
%waitcount%
%TPCH_AUTOMATION_EXEC%\semaphore -release %TPCH_Sentinel%
exit
endlocal
```

Appendix E: Implementation Specific Layer and Source Code

cArrConstraints.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cArrConstraints"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:      cArrConstraints.cls
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cConstraint objects.
'             Type-safe wrapper around cNodeCollections.
'             Also contains additional functions that determine all the
'             constraints for a step, all constraints in a workspace,
'             validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrConstraints As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cArrConstraints."
Public Sub SaveWspConstraints(ByVal lngWorkspace As Long)
   ' Calls a procedure to commit all changes to the constraints
   ' in the passed in workspace.

   Call mcarrConstraints.Save(lngWorkspace)

End Sub
Public Property Set ConstraintDB(vdata As Database)

   Set mcarrConstraints.NodeDB = vdata

End Property
Public Property Get ConstraintDB() As Database

   Set ConstraintDB = mcarrConstraints.NodeDB

End Property

Public Sub Modify(cConsToUpdate As cConstraint)

   ' Modify the constraint record
   Call mcarrConstraints.Modify(cConsToUpdate)

End Sub
Public Sub CreateNewConstraintVersion(ByVal lngStepId As Long, _
      ByVal strNewVersion As String, _
      ByVal strOldVersion As String, _
      ByVal intStepType As Integer)

   ' Does all the processing needed to create new versions of
   ' all the constraints for a given step
   ' It inserts new constraint records in the database with
   ' the new version numbers on them
   ' It also updates the version number on all constraints
   ' for the step in the array to the new version passed in
   ' Since it handles both global and manager/worker steps,
   ' it checks for the step_id or global_step_id fields,
   ' depending on the type of step

   Dim lngIndex As Long
   Dim cUpdateConstraint As cConstraint

   On Error GoTo CreateNewConstraintVersionErr
   mstrSource = mstrModuleName & "CreateNewConstraintVersion"

   ' Update the version/global version on Constraint with the
   ' passed in step/global step id
   For lngIndex = 0 To mcarrConstraints.Count - 1
      Set cUpdateConstraint = mcarrConstraints(lngIndex)
      If intStepType = gintGlobalStep Then
         If cUpdateConstraint.GlobalStepId = lngStepId And _
               cUpdateConstraint.IndOperation <> DeleteOp Then
            cUpdateConstraint.GlobalVersionNo = strNewVersion

            ' Set the operation to indicate an insert
            cUpdateConstraint.IndOperation = InsertOp
         End If
      Else
         If cUpdateConstraint.StepId = lngStepId And _
               cUpdateConstraint.IndOperation <> DeleteOp Then
            cUpdateConstraint.VersionNo = strNewVersion

            ' Set the operation to indicate an insert
            cUpdateConstraint.IndOperation = InsertOp
         End If
      End If
   Next lngIndex

   Exit Sub

CreateNewConstraintVersionErr:
   LogErrors Errors
   gstrSource = mstrModuleName & "CreateNewConstraintVersion"
   On Error GoTo 0
   Err.Raise vbObjectError + errCreateNewConstraintVersionFailed, _
         mstrSource, _
         LoadResString(errCreateNewConstraintVersionFailed)

End Sub
Private Sub Class_Initialize()

   Set mcarrConstraints = New cNodeCollections
   BugMessage "cArrConstraints: Initialize event - setting Constraint count to 0"

End Sub

Private Sub Class_Terminate()

   Set mcarrConstraints = Nothing
   BugMessage "cArrConstraints: Terminate event triggered"

End Sub

Public Sub Add(ByVal cConstraintToAdd As cConstraint)

   Set cConstraintToAdd.NodeDB = mcarrConstraints.NodeDB

   ' Retrieve a unique constraint identifier
   cConstraintToAdd.ConstraintId = cConstraintToAdd.NextIdentifier

   ' Call a procedure to load the constraint record in the array
   Call mcarrConstraints.Add(cConstraintToAdd)

End Sub
Public Sub Delete(ByVal cOldConstraint As cConstraint)

   Dim lngDeleteElement As Long
   Dim cConsToDelete As cConstraint

   lngDeleteElement = QueryConstraintIndex(cOldConstraint.ConstraintId)
   Set cConsToDelete = mcarrConstraints(lngDeleteElement)
```

```
    Call mcarrConstraints.Delete(cConsToDelete.Position)

    Set cConsToDelete = Nothing

End Sub
Private Function QueryConstraintIndex(lngConstraintId As Long) _
        As Long

    Dim lngIndex As Integer

    ' Find the element in the array to be deleted
    For lngIndex = 0 To mcarrConstraints.Count - 1

        ' Note: The constraint id is not a primary key field in
        ' the database - there can be multiple records with the
        ' same constraint_id but for different versions of a step
        ' However, since we'll always load the constraint information
        ' for the latest version of a step, we'll have just one
        ' constraint record with a given constraint_id
        If mcarrConstraints(lngIndex).ConstraintId = lngConstraintId Then
            QueryConstraintIndex = lngIndex
            Exit Function
        End If

    Next lngIndex

    ' Raise error that Constraint has not been found
    ShowError errConstraintNotFound
    On Error GoTo 0
    Err.Raise vbObjectError + errConstraintNotFound, mstrSource, _
        LoadResString(errConstraintNotFound)

End Function

Public Function QueryConstraint(ByVal lngConstraintId As Long) _
    As cConstraint

    ' Returns a cConstraint object with the property values
    ' corresponding to the Constraint Identifier, lngConstraintId

    Dim lngQueryElement As Long

    lngQueryElement = QueryConstraintIndex(lngConstraintId)

    ' Set the return value to the queried Constraint
    Set QueryConstraint = mcarrConstraints(lngQueryElement)

End Function

Public Sub LoadConstraints(ByVal lngWorkspaceId As Long, rstStepsInWsp As
Recordset)

    ' Loads the constraints array with all the constraints
    ' for the workspace
    Dim recConstraints As Recordset
    Dim qyCons As DAO.QueryDef
    Dim strSql As String
    Dim dtStart As Date

    On Error GoTo LoadConstraintsErr
    mstrSource = mstrModuleName & "LoadConstraints"

    If rstStepsInWsp.RecordCount = 0 Then
        Exit Sub
    End If

    ' First check if the database object has been set
    If mcarrConstraints.NodeDB Is Nothing Then
        On Error GoTo 0
        Err.Raise vbObjectError + errSetDBBeforeLoad, _
            mstrSource, _
            LoadResString(errSetDBBeforeLoad)
    End If

    dtStart = Now

    ' Select based on the global step id since there might
    ' be constraints for a global step that run are executed
    ' for the workspace
    ' This method has the advantage that if the steps are queried right, everything
else follows
    strSql = "Select a.constraint_id, a.step_id, a.version_no, " & _
        " a.constraint_type, a.global_step_id, a.global_version_no, " & _
        " a.sequence_no, b.workspace_id " & _
        " from step_constraints a, att_steps b " & _
        " where a.global_step_id = b.step_id " & _
        " and a.global_version_no = b.version_no " & _
        " and a.global_step_id = [g_s_id] " & _
        " and a.global_version_no = [g_ver_no] " & _
        " and b.archived_flag = [archived] "

    ' Find the highest X-component of the version number
    strSql = strSql & " AND ( a.step_id = 0 or ( cint( mid( a.version_no, 1, instr(
a.version_no, " & gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS d " & _
        " WHERE a.step_id = d.step_id " & _
        " and d.archived_flag = [archived] ) "

    ' Find the highest Y-component of the version number for the highest
X-component
    strSql = strSql & " AND cint( mid( a.version_no, instr( a.version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) + 1 ) ) = " & _
        " ( select max( cint( mid( version_no, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) + 1 ) ) ) " & _
        " from att_steps AS y " & _
        " Where a.step_id = y.step_id " & _
        " AND cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS c " & _
        " WHERE y.step_id = c.step_id " & _
        " and c.archived_flag = [archived] ) ) ) ) "

    ' Order the constraints by sequence within a given step
    strSql = strSql & " order by a.sequence_no "

    Set qyCons = mcarrConstraints.NodeDB.CreateQueryDef(gstrEmptyString,
strSql)
    qyCons.Parameters("archived").Value = False

    rstStepsInWsp.MoveFirst

    While Not rstStepsInWsp.EOF

        If Not (rstStepsInWsp!global_flag) Then
            qyCons.Close
            BugMessage "Query constraints Read + load took: " & CStr(DateDiff("s",
dtStart, Now))
            Exit Sub
        End If

        qyCons.Parameters("g_s_id").Value = rstStepsInWsp!step_id
        qyCons.Parameters("g_ver_no").Value = rstStepsInWsp!version_no

        Set recConstraints = qyCons.OpenRecordset(dbOpenSnapshot)

        Call LoadRecordsetInConstraintArray(recConstraints)
        recConstraints.Close

        rstStepsInWsp.MoveNext
    Wend

    qyCons.Close
```

```
   BugMessage "Query constraints Read + load took: " & CStr(DateDiff("s",
dtStart, Now))

   Exit Sub

LoadConstraintsErr:
   LogErrors Errors
   gstrSource = mstrModuleName & "LoadConstraints"
   On Error GoTo 0
   Err.Raise vbObjectError + errLoadDataFailed, _
         mstrSource, _
         LoadResString(errLoadDataFailed)

End Sub
Public Sub UnloadStepConstraints(ByVal lngStepId As Long)

   ' Unloads all the constraints for the workspace from
   ' the constraints array

   Dim lngIndex As Long

   ' Find all constraints in the array with a matching step id
   ' It is important to step in reverse order through the array,
   ' since we delete constraint records!
   For lngIndex = mcarrConstraints.Count - 1 To 0 Step -1
      If mcarrConstraints(lngIndex).GlobalStepId = lngStepId Then

         ' Unload the constraint from the array
         Call mcarrConstraints.Unload(lngIndex)

      End If
   Next lngIndex

End Sub
Public Sub UnloadConstraint(cOldConstraint As cConstraint)
   ' Unloads the constraint from the constraints array

   Dim lngDeleteElement As Long

   lngDeleteElement = QueryConstraintIndex(cOldConstraint.ConstraintId)

   Call mcarrConstraints.Unload(lngDeleteElement)

End Sub
Private Sub LoadRecordsetInConstraintArray(ByVal recConstraints As
Recordset)
   ' Loads all the constraint records in the passed in
   ' recordset into the array

   Dim cNewConstraint As cConstraint

   On Error GoTo LoadRecordsetInConsArrayErr
   mstrSource = mstrModuleName & "LoadRecordsetInConstraintArray"

   If recConstraints.RecordCount = 0 Then
      Exit Sub
   End If

   recConstraints.MoveFirst
   While Not recConstraints.EOF
      Set cNewConstraint = New cConstraint

      ' Initialize Constraint values
      cNewConstraint.ConstraintId = CLng(ErrorOnNullField(recConstraints,
"Constraint_id"))
      cNewConstraint.StepId = CLng(ErrorOnNullField(recConstraints,
"step_id"))
      cNewConstraint.VersionNo = CStr(ErrorOnNullField(recConstraints,
"version_no"))

      cNewConstraint.GlobalStepId = CLng(ErrorOnNullField(recConstraints,
"global_step_id"))
      cNewConstraint.GlobalVersionNo = CStr(ErrorOnNullField(recConstraints,
"global_version_no"))
      cNewConstraint.SequenceNo = CInt(ErrorOnNullField(recConstraints,
"sequence_no"))

      cNewConstraint.WorkspaceId = CLng(ErrorOnNullField(recConstraints,
FLD_ID_WORKSPACE))
      cNewConstraint.ConstraintType = CInt(ErrorOnNullField(recConstraints,
"constraint_type"))

      ' Add this record to the array of Constraints
      mcarrConstraints.Load cNewConstraint

      Set cNewConstraint = Nothing
      recConstraints.MoveNext
   Wend

   Exit Sub

LoadRecordsetInConsArrayErr:
   LogErrors Errors
   gstrSource = mstrModuleName & "LoadRecordsetInConstraintArray"
   On Error GoTo 0
   Err.Raise vbObjectError + errLoadRsInArrayFailed, _
         mstrSource, _
         LoadResString(errLoadRsInArrayFailed)

End Sub

Public Function ConstraintsForStep( _
      ByVal lngStepId As Long, _
      ByVal strVersionNo As String, _
      Optional ByVal intConstraintType As ConstraintType = 0, _
      Optional ByVal blnSort As Boolean = True, _
      Optional ByVal blnGlobal As Boolean = False, _
      Optional ByVal blnGlobalConstraintsOnly As Boolean = False) _
      As Variant

   ' Returns a variant containing an array of cConstraint objects,
   ' containing all the constraints that have been defined for the
   ' given step. If the Global flag is set to true, the
   ' search will be made for all the constraints that have
   ' a matching global_step_id

   Dim lngIndex As Long
   Dim cStepConstraint() As cConstraint
   Dim lngConstraintCount As Long
   Dim cTempConstraint As cConstraint

   On Error GoTo ConstraintsForStepErr
   mstrSource = mstrModuleName & "ConstraintsForStep"

   lngConstraintCount = 0

   ' Find each element in the constraints array
   For lngIndex = 0 To mcarrConstraints.Count - 1
      ' If a constraint type has been specified then check
      ' if the constraint type for the record matches the
      ' passed in type
      Set cTempConstraint = mcarrConstraints(lngIndex)
      If Not blnGlobal Then
         If cTempConstraint.StepId = lngStepId And _
               cTempConstraint.VersionNo = strVersionNo And _
               cTempConstraint.IndOperation <> DeleteOp And _
               (intConstraintType = 0 Or _
               cTempConstraint.ConstraintType = intConstraintType) Then
            ' We have a matching constraint for the given step
            AddArrayElement cStepConstraint, _
                  cTempConstraint, lngConstraintCount
         End If
      Else
         If cTempConstraint.GlobalStepId = lngStepId And _
               cTempConstraint.GlobalVersionNo = strVersionNo And _
               cTempConstraint.IndOperation <> DeleteOp Then
            If blnGlobalConstraintsOnly = False Or _
                  (blnGlobalConstraintsOnly And _
```

```
                cTempConstraint.StepId = 0 And _
                cTempConstraint.VersionNo = gstrMinVersion) Then

            ' We have a matching constraint for the global step
            AddArrayElement cStepConstraint, _
                cTempConstraint, lngConstraintCount
        End If
      End If
    End If

  Next lngIndex

  ' Set the return value of the function to the array of
  ' constraints that has been built above
  If lngConstraintCount = 0 Then
    ConstraintsForStep = Empty
  Else
    ConstraintsForStep = cStepConstraint()
  End If

  ' Sort the constraints
  If blnSort Then
    Call QuickSort(ConstraintsForStep)
  End If

  Exit Function

ConstraintsForStepErr:
  LogErrors Errors
  On Error GoTo 0
  Err.Raise vbObjectError + errConstraintsForStepFailed, _
      mstrSource, _
      LoadResString(errConstraintsForStepFailed)

End Function
Private Sub AddArrayElement(ByRef arrNodes() As cConstraint, _
    ByVal objToAdd As cConstraint, _
    ByRef lngCount As Long)
  ' Adds the passed in object to the array

  ' Increase the array dimension and add the object to it
  ReDim Preserve arrNodes(lngCount)
  Set arrNodes(lngCount) = objToAdd
  lngCount = lngCount + 1

End Sub

Public Function ConstraintsForWsp( _
    ByVal lngWorkspaceId As Long, _
    Optional ByVal intConstraintType As Integer = 0, _
    Optional ByVal blnSort As Boolean = True, _
    Optional ByVal blnGlobalConstraintsOnly As Boolean = False) _
    As Variant

  ' Returns a variant containing an array of cConstraint objects,
  ' containing all the constraints that have been defined for the
  ' given workspace.

  Dim lngIndex As Long
  Dim cWspConstraint() As cConstraint
  Dim lngConstraintCount As Long
  Dim cTempConstraint As cConstraint

  On Error GoTo ConstraintsForWspErr
  mstrSource = mstrModuleName & "ConstraintsForWsp"

  lngConstraintCount = 0

  ' Find each element in the constraints array
  For lngIndex = 0 To mcarrConstraints.Count - 1
    ' If a constraint type has been specified then check
    ' if the constraint type for the record matches the
    ' passed in type
    Set cTempConstraint = mcarrConstraints(lngIndex)
    If cTempConstraint.WorkspaceId = lngWorkspaceId And _
        cTempConstraint.IndOperation <> DeleteOp And _
        (intConstraintType = 0 Or _
        cTempConstraint.ConstraintType = intConstraintType) Then

      If blnGlobalConstraintsOnly = False Or _
          (blnGlobalConstraintsOnly And _
          cTempConstraint.StepId = 0 And _
          cTempConstraint.VersionNo = gstrMinVersion) Then

        ' We have a matching constraint for the workspace
        AddArrayElement cWspConstraint, _
            cTempConstraint, lngConstraintCount
      End If
    End If
  Next lngIndex

  ' Set the return value of the function to the array of
  ' constraints that has been built above
  If lngConstraintCount = 0 Then
    ConstraintsForWsp = Empty
  Else
    ConstraintsForWsp = cWspConstraint()
  End If

  ' Sort the constraints
  If blnSort Then
    Call QuickSort(ConstraintsForWsp)
  End If

  Exit Function

ConstraintsForWspErr:
  LogErrors Errors
  On Error GoTo 0
  Err.Raise vbObjectError + errConstraintsForWspFailed, _
      mstrSource, _
      LoadResString(errConstraintsForWspFailed)

End Function
Public Function PreConstraintsForStep( _
    ByVal lngStepId As Long, _
    ByVal strVersionNo As String, _
    Optional ByVal blnSort As Boolean) As Variant

  ' Returns a variant containing an array of cConstraint objects,
  ' containing all the pre-execution constraints that have
  ' been defined for the given step_id and version

  ' Call a function that will return a variant containing
  ' all the constraints of the passed in type
  PreConstraintsForStep = ConstraintsForStep(lngStepId, _
      strVersionNo, gintPreStep, blnSort)

End Function
Public Function PostConstraintsForStep( _
    ByVal lngStepId As Long, _
    ByVal strVersionNo As String, _
    Optional ByVal blnSort As Boolean) As Variant

  ' Returns a variant containing an array of cConstraint objects,
  ' containing all the Post-execution constraints that have
  ' been defined for the given step_id and version

  ' Call a function that will return a variant containing
  ' all the constraints of the passed in type
  PostConstraintsForStep = ConstraintsForStep(lngStepId, _
      strVersionNo, gintPostStep, blnSort)

End Function
Public Function PostConstraintsForWsp( _
    ByVal lngWorkspaceId As Long, _
    Optional ByVal blnSort As Boolean) As Variant
```

```
    ' Returns a variant containing an array of cConstraint objects,
    ' containing all the Post-execution globals that have
    ' been defined for the workspace

    ' Call a function that will return a variant containing
    ' all the constraints of the passed in type
    PostConstraintsForWsp = ConstraintsForWsp(lngWorkspaceId, _
            gintPostStep, blnSort, True)

End Function
Public Function PreConstraintsForWsp( _
        ByVal lngWorkspaceId As Long, _
        Optional ByVal blnSort As Boolean) As Variant

    ' Returns a variant containing an array of cConstraint objects,
    ' containing all the Pre-execution globals that have
    ' been defined for the workspace

    ' Call a function that will return a variant containing
    ' all the constraints of the passed in type
    PreConstraintsForWsp = ConstraintsForWsp(lngWorkspaceId, _
            gintPreStep, blnSort, True)

End Function
Public Property Get ConstraintCount() As Long

    ConstraintCount = mcarrConstraints.Count

End Property
```

cArrParameters.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cArrParameters"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:      cArrParameters.cls
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cParameter objects.
'             Type-safe wrapper around cNodeCollections.
'             Also contains additional functions to determine parameter
'             values, validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrParameters As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cArrParameters."

Public Sub InitBuiltInsForRun(lWspId As Long, lRunId As Long)

    Dim cParamRec As cParameter

    ' Initialize the values of the run_id and output_dir built-in parameters and save
them
    ' to the database
    Set cParamRec = GetParameterValue(lWspId, PARAM_RUN_ID)
    cParamRec.ParameterValue = CStr(lRunId)
    Call Modify(cParamRec)

    Set cParamRec = GetParameterValue(lWspId, PARAM_OUTPUT_DIR)
    cParamRec.ParameterValue = GetDefaultDir(lWspId, Me)
    cParamRec.ParameterValue = cParamRec.ParameterValue & gstrFileSeparator
& CStr(lRunId)
    Call Modify(cParamRec)

    Call SaveParametersInWsp(lWspId)

End Sub


Public Property Set ParamDatabase(vdata As Database)

    Set mcarrParameters.NodeDB = vdata

End Property
Public Sub Modify(cModifiedParam As cParameter)

    ' First check if the parameter record is valid
    Call CheckDupParamName(cModifiedParam)

    Call mcarrParameters.Modify(cModifiedParam)

End Sub
Public Sub Load(ByRef cParamToAdd As cParameter)

    Call mcarrParameters.Load(cParamToAdd)

End Sub
Public Sub Add(ByRef cParamToAdd As cParameter)

    Set cParamToAdd.NodeDB = mcarrParameters.NodeDB

    ' First check if the parameter record is valid
    Call Validate(cParamToAdd)

    ' Retrieve a unique parameter identifier
    cParamToAdd.ParameterId = cParamToAdd.NextIdentifier

    Call mcarrParameters.Add(cParamToAdd)

End Sub

Public Sub Unload(lngParamToDelete As Long)

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lngParamToDelete)

    Call mcarrParameters.Unload(lngDeleteElement)

End Sub

Public Sub SaveParametersInWsp(ByVal lngWorkspace As Long)
    ' Calls a procedure to commit all changes to the parameters
    ' for the passed in workspace.

    ' Call a procedure to save all parameter records for the
    ' workspace
    Call mcarrParameters.Save(lngWorkspace)

End Sub
Public Function GetParameterValue(ByVal lngWorkspace As Long, _
        ByVal strParamName As String) As cParameter
    ' Returns the value for the passed in workspace parameter

    Dim cParamRec As cParameter
    Dim lngIndex As Long

    On Error GoTo GetParameterValueErr

    ' Find all parameters in the array with a matching workspace id
    For lngIndex = 0 To mcarrParameters.Count - 1
```

```
      Set cParamRec = mcarrParameters(lngIndex)
      If cParamRec.WorkspaceId = lngWorkspace And _
            cParamRec.ParameterName = strParamName Then

         Set GetParameterValue = cParamRec
         Exit For
      End If
   Next lngIndex

   If lngIndex > mcarrParameters.Count - 1 Then
      ' The parameter has not been defined for the workspace
      ' Raise an error
      On Error GoTo 0
      Err.Raise vbObjectError + errParamNameInvalid, _
            mstrModuleName & "GetParameterValue", _
            LoadResString(errParamNameInvalid)
   End If

   Exit Function

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

End Function
Public Sub Delete(lngParamToDelete As Long)
   ' Delete the passed in parameter

   Dim lngDeleteElement As Long

   lngDeleteElement = QueryIndex(lngParamToDelete)
   Call mcarrParameters.Delete(lngDeleteElement)

End Sub
Private Function QueryIndex(lngParameterId As Long) As Long

   Dim lngIndex As Long

   ' Find the matching parameter record in the array
   For lngIndex = 0 To mcarrParameters.Count - 1
      If mcarrParameters(lngIndex).ParameterId = lngParameterId And _
            mcarrParameters(lngIndex).IndOperation <> DeleteOp Then
         QueryIndex = lngIndex
         Exit Function
      End If
   Next lngIndex

   ' Raise error that parameter has not been found
   On Error GoTo 0
   Err.Raise vbObjectError + errParamNotFound, "cArrParameters.QueryIndex",
_
      LoadResString(errParamNotFound)

End Function

Public Function QueryParameter(lngParameterId As Long) _
      As cParameter

   Dim lngQueryElement As Long

   lngQueryElement = QueryIndex(lngParameterId)

   ' Return the queried parameter object
   Set QueryParameter = mcarrParameters(lngQueryElement)

End Function
Public Property Get ParameterCount() As Long

   ParameterCount = mcarrParameters.Count

End Property
Public Property Get Item(lngIndex As Long) As cParameter
Attribute Item.VB_UserMemId = 0

   Set Item = mcarrParameters(lngIndex)

End Property

Public Sub Validate(ByVal cParamToValidate As cParameter)
   ' This procedure is necessary since the class cannot validate
   ' all the parameter properties on it's own. This is 'coz we
   ' might have created new parameters in the workspace, but not
   ' saved them to the database yet - hence the duplicate check
   ' has to be repeated in the array

   Dim lngIndex As Long
   Dim cTempParam As cParameter

   On Error GoTo ValidateErr

   ' Check if the parameter name already exists in the workspace
   For lngIndex = 0 To mcarrParameters.Count - 1
      Set cTempParam = mcarrParameters(lngIndex)
      If cTempParam.WorkspaceId = cParamToValidate.WorkspaceId And _
            cTempParam.ParameterName = cParamToValidate.ParameterName
And _
            cTempParam.IndOperation <> DeleteOp Then
         On Error GoTo 0
         Err.Raise vbObjectError + errDuplicateParameterName, _
            mstrSource, LoadResString(errDuplicateParameterName)
      End If
   Next lngIndex

   Exit Sub

ValidateErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Validate"
   On Error GoTo 0
   Err.Raise vbObjectError + errValidateFailed, _
      mstrSource, LoadResString(errValidateFailed)

End Sub
Public Sub CheckDupParamName(ByVal cParamToValidate As cParameter)

   Dim lngIndex As Long
   Dim cTempParam As cParameter

   ' Check if the parameter name already exists in the workspace
   For lngIndex = 0 To mcarrParameters.Count - 1
      Set cTempParam = mcarrParameters(lngIndex)
      If cTempParam.WorkspaceId = cParamToValidate.WorkspaceId And _
            cTempParam.ParameterName = cParamToValidate.ParameterName
And _
            cTempParam.ParameterId <> cParamToValidate.ParameterId And _
            cTempParam.IndOperation <> DeleteOp Then
         ShowError errDuplicateParameterName
         On Error GoTo 0
         Err.Raise vbObjectError + errDuplicateParameterName, _
            mstrSource, LoadResString(errDuplicateParameterName)
      End If
   Next lngIndex

End Sub

Private Sub Class_Initialize()

   'bugmessage "cArrParameters: Initialize event - setting parameter count to 0"
   Set mcarrParameters = New cNodeCollections

End Sub
```

```
Private Sub Class_Terminate()

   Set mcarrParameters = Nothing

End Sub
```

cArrSteps.cls

```
 VERSION 1.0 CLASS

BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cArrSteps"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:      cArrSteps.cls
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cStep objects.
'             Type-safe wrapper around cNodeCollections.
'             Also contains additional functions to update parent version
'             on substeps, validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrSteps As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cArrSteps."

Public Sub Unload(lngStepToDelete As Long)

   Dim lngDeleteElement As Long
   Dim cUnloadStep As cStep

   lngDeleteElement = QueryStepIndex(lngStepToDelete)
   Set cUnloadStep = QueryStep(lngStepToDelete)

   ' First unload all iterators for the step
   Call cUnloadStep.UnloadIterators

   ' Unload the step from the collection
   Call mcarrSteps.Unload(lngDeleteElement)

End Sub
Public Sub Modify(cModifiedStep As cStep)

   Dim iAppend As Integer
   Dim sLabel As String

   On Error GoTo ModifyErr

   iAppend = 0
   sLabel = cModifiedStep.StepLabel

   Validate cModifiedStep

   Call mcarrSteps.Modify(cModifiedStep)

   Exit Sub

ModifyErr:
   ' If the error raised by the add function is due to a duplication
   ' of the step label, then try to generate a unique label
   If Err.Number - vbObjectError = errStepLabelUnique Then
      iAppend = iAppend + 1
      cModifiedStep.StepLabel = sLabel & CStr(iAppend)
      ' Try to insert the step record again
      Resume
   End If

   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errModifyStepFailed, _
         gstrSource, _
         LoadResString(errModifyStepFailed)

End Sub
Public Sub UpdateParentVersion(ByVal lngStepId As Long, _
      ByVal strNewVersion As String, _
      ByVal strOldVersion As String, _
      ByVal intStepType As Integer)

   ' Does all the processing needed to update the parent version
   ' number on all the sub-steps for a given step
   ' It updates the parent version no in the database for all
   ' sub-steps of the passed in step id
   ' It also updates the parent version number on all sub-steps
   ' in the array to the new version passed in

   Dim lngIndex As Long
   Dim cUpdateStep As cStep

   On Error GoTo UpdateParentVersionErr

   If intStepType <> gintManagerStep Then
      ' Only a manager can have sub-steps - if the passed
      ' in step is not a manager, exit
      Exit Sub
   End If

   ' For all steps in the array
   For lngIndex = 0 To mcarrSteps.Count - 1

      Set cUpdateStep = mcarrSteps(lngIndex)

      ' If the current step is a sub-step of the passed in step
      If cUpdateStep.ParentStepId = lngStepId And _
            cUpdateStep.ParentVersionNo = strOldVersion And _
            Not cUpdateStep.ArchivedFlag Then

         ' Update the parent version number for the sub-step
         ' in the array
         cUpdateStep.ParentVersionNo = strNewVersion

         ' Update the parent version number for the sub-step
         ' in the array
         Call Modify(cUpdateStep)

      End If
   Next lngIndex

   Exit Sub

UpdateParentVersionErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "UpdateParentVersion"
   On Error GoTo 0
   Err.Raise vbObjectError + errUpdateParentVersionFailed, _
         mstrSource, _
         LoadResString(errUpdateParentVersionFailed)

End Sub
Private Sub Validate(cCheckStep As cStep)

   ' Step validations that depend on other steps in the collection
```

```
    Dim lngIndex As Long

    ' Ensure that the step label is unique in the workspace
    For lngIndex = 0 To mcarrSteps.Count - 1

        ' If the current step is a sub-step of the passed in step
        If mcarrSteps(lngIndex).WorkspaceId = cCheckStep.WorkspaceId And _
              mcarrSteps(lngIndex).StepLabel = cCheckStep.StepLabel And _
              mcarrSteps(lngIndex).StepId <> cCheckStep.StepId And _
              mcarrSteps(lngIndex).IndOperation <> DeleteOp Then
            On Error GoTo 0
            Err.Raise vbObjectError + errStepLabelUnique, _
                  mstrModuleName & "Validate", _
                  LoadResString(errStepLabelUnique)
        End If
    Next lngIndex

End Sub

Public Sub ValidateStep(cCheckStep As cStep)

    On Error GoTo ValidateStepErr

    'Public wrapper for Validate function (2 many Validates)
    Call Validate(cCheckStep)

    Exit Sub

ValidateStepErr:
    ShowError errStepLabelUnique
    On Error GoTo 0
    Err.Raise vbObjectError + errValidateFailed, _
          gstrSource, _
          LoadResString(errValidateFailed)
End Sub

Private Sub Class_Initialize()

    BugMessage "cArrSteps: Initialize event - setting step count to 0"
    Set mcarrSteps = New cNodeCollections

End Sub

Private Sub Class_Terminate()

    BugMessage "cArrSteps: Terminate event triggered"
    Set mcarrSteps = Nothing

End Sub

Public Sub Add(ByVal cStepToAdd As cStep)

    Dim iAppend As Integer
    Dim sLabel As String

    On Error GoTo AddErr

    iAppend = 0
    sLabel = cStepToAdd.StepLabel

    Set cStepToAdd.NodeDB = mcarrSteps.NodeDB

    ' Retrieve a unique step identifier
    cStepToAdd.StepId = cStepToAdd.NextStepId

    Validate cStepToAdd

    ' Call a procedure to add the step record
    Call mcarrSteps.Add(cStepToAdd)

    ' Call a procedure to add all iterators for the step
    cStepToAdd.AddAllIterators

    Exit Sub

AddErr:
    ' If the error raised by the add function is due to a duplication
    ' of the step label, then try to generate a unique label
    If Err.Number - vbObjectError = errStepLabelUnique Then
        iAppend = iAppend + 1
        cStepToAdd.StepLabel = sLabel & CStr(iAppend)
        ' Try to insert the step record again
        Resume
    End If

    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertStepFailed, _
          gstrSource, _
          LoadResString(errInsertStepFailed)

End Sub
Public Sub Load(cStepToLoad As cStep)

    Call mcarrSteps.Load(cStepToLoad)

End Sub
Public Sub SaveStepsInWsp(ByVal lngWorkspace As Long)
    ' Calls a procedure to commit all changes to the steps
    ' in the passed in workspace.

    Dim lngIndex As Integer

    ' Find all steps in the array with a matching workspace id
    ' It is important to step in reverse order through the array,
    ' since we delete step records sometimes!
    For lngIndex = mcarrSteps.Count - 1 To 0 Step -1
        If mcarrSteps(lngIndex).WorkspaceId = lngWorkspace Then

            ' Call a procedure to commit all changes to the
            ' Step record, if any
            Call CommitStep(mcarrSteps(lngIndex), lngIndex)

        End If
    Next lngIndex

End Sub
Private Sub CommitStep(ByVal cCommitStep As cStep, _
        ByVal intIndex As Integer)
    ' This procedure checks if any changes have been made to the
    ' passed in Step. If so, it calls the step methods to commit
    ' the changes.

    ' First commit all changes to the iterator records for
    ' the step
    cCommitStep.SaveIterators

    Call mcarrSteps.Commit(cCommitStep, intIndex)

End Sub
Public Sub Delete(lngStepToDelete As Long)

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryStepIndex(lngStepToDelete)
    Call mcarrSteps.Delete(lngDeleteElement)

End Sub
Public Function QueryStepIndex(lngStepId As Long) As Long

    Dim lngIndex As Long

    ' Find the element in the array that corresponds to the
    ' passed in step id - note that while there will be multiple
    ' versions of a step in the database, only one version will
    ' be currently loaded in the array - meaning that the stepid
    ' is enough to uniquely identify a step
```

```
    For lngIndex = 0 To mcarrSteps.Count - 1
        If mcarrSteps(lngIndex).StepId = lngStepId Then
            QueryStepIndex = lngIndex
            Exit Function
        End If
    Next lngIndex

    ' Raise error that step has not been found
    On Error GoTo 0
    Err.Raise vbObjectError + errStepNotFound, mstrSource, _
        LoadResString(errStepNotFound)

End Function

Public Function QueryStep(ByVal lngStepId As Long) As cStep

    ' Populates the passed in cStep object with the property
    ' values corresponding to the Step Identifier, lngStepId

    Dim lngQueryElement As Integer

    lngQueryElement = QueryStepIndex(lngStepId)

    ' Initialize the passed in step object to the queried step
    Set QueryStep = mcarrSteps(lngQueryElement)

End Function
Public Property Get Item(ByVal Position As Long) As cStep
Attribute Item.VB_UserMemId = 0

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mcarrSteps.Count Then
        Set Item = mcarrSteps(Position)
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Set Item(ByVal Position As Long, _
        ByVal cStepRec As cStep)

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mcarrSteps.Count Then
        Set mcarrSteps(Position) = cStepRec
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Set StepDB(vdata As Database)

    Set mcarrSteps.NodeDB = vdata

End Property
Public Function SubSteps(ByVal lngStepId As Long, _
        ByVal strVersionNo As String) As Variant

    ' Returns a variant containing an array of all the substeps
    ' for the passed in step

    Dim intIndex As Integer
    Dim cSubSteps() As cStep
    Dim lngStepCount As Long
    Dim cQueryStep As cStep

    On Error GoTo SubStepsErr

    lngStepCount = 0

    Set cQueryStep = QueryStep(lngStepId)

    ' Only a manager can have sub-steps
    If cQueryStep.StepType = gintManagerStep Then

        ' For each element in the Steps array
        For intIndex = 0 To mcarrSteps.Count - 1
            ' Check if the parent step id and parent version number
            ' match the passed in step
            If mcarrSteps(intIndex).ParentStepId = lngStepId And _
                    mcarrSteps(intIndex).ParentVersionNo = strVersionNo And _
                    mcarrSteps(intIndex).IndOperation <> DeleteOp Then

                ' Increase the array dimension and add the step
                ' to it
                ReDim Preserve cSubSteps(lngStepCount)
                Set cSubSteps(lngStepCount) = mcarrSteps(intIndex)
                lngStepCount = lngStepCount + 1

            End If
        Next intIndex

    End If

    ' Set the return value of the function to the array of
    ' Steps that has been built above
    If lngStepCount = 0 Then
        SubSteps = Empty
    Else
        SubSteps = cSubSteps()
    End If

    Exit Function

SubStepsErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "SubSteps"
    On Error GoTo 0
    Err.Raise vbObjectError + errSubStepsFailed, _
            mstrSource, _
            LoadResString(errSubStepsFailed)

End Function

Public Property Get StepCount() As Integer

    StepCount = mcarrSteps.Count

End Property
```

cAsyncShell.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cAsyncShell"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'-----------------------------------------------------------------------------
' Copyright © 1997 Microsoft Corporation. All rights reserved.
'
' You have a royalty-free right to use, modify, reproduce and distribute the
' Sample Application Files (and/or any modified version) in any way you find
' useful, provided that you agree that Microsoft has no warranty, obligations or
' liability for any Sample Application Files.
'-----------------------------------------------------------------------------

Option Explicit

' Used to indicate the source module name when errors
```

```
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cAsyncShell."

Public Event Terminated()

Private WithEvents moTimer As cTimerSM
Attribute moTimer.VB_VarHelpID = -1
Private proc As PROCESS_INFORMATION
Private mfShelling As Boolean


'-----------------------------------------------------------------------------
'Initialization and cleanup:

Private Sub Class_Initialize()
   Set moTimer = New cTimerSM
End Sub

Private Sub Class_Terminate()
   If mfShelling Then CloseHandle proc.hProcess
End Sub

'-----------------------------------------------------------------------------
'Shelling:

Public Sub Shell(CommandLine As String, Optional PollingInterval As Long =
1000)
   Dim Start As STARTUPINFO

   If mfShelling Then
      On Error GoTo 0
      Err.Raise vbObjectError + errInstanceInUse, _
            mstrSource, _
            LoadResString(errInstanceInUse)
   End If
   mfShelling = True

   ' Initialize the STARTUPINFO structure:
   Start.cb = Len(Start)
   Start.dwFlags = STARTF_USESHOWWINDOW
   Start.wShowWindow = SW_SHOWMINNOACTIVE

   ' Start the shelled application:
   CreateProcessA 0&, CommandLine, 0&, 0&, 1&, _
      NORMAL_PRIORITY_CLASS, 0&, 0&, Start, proc

   With moTimer
      If PollingInterval > 0 Then
         .Interval = PollingInterval
      Else
         .Interval = 1000
      End If
      .Enabled = True
   End With
End Sub
'-----------------------------------------------------------------------------
'Aborting:
Public Sub Abort()
   Dim nCode As Long
'    Dim X As Integer
'    Dim ReturnVal As Integer

   On Error GoTo AbortErr

   If Not mfShelling Then
      Call WriteError(errProgramError, mstrSource)
   Else
'       If IsWindow(proc.hProcess) = False Then Exit Sub
'
'       If (GetWindowLong(proc.hProcess, GWL_STYLE) And WS_DISABLED)
Then Exit Sub
'
'       If IsWindow(proc.hProcess) Then
'           If Not (GetWindowLong(proc.hProcess, GWL_STYLE) And
WS_DISABLED) Then
'               X = PostMessage(proc.hProcess, WM_CANCELMODE, 0, 0&)
'               X = PostMessage(proc.hProcess, WM_CLOSE, 0, 0&)
'           End If
'       End If

      If TerminateProcess(proc.hProcess, 0&) = 0 Then
         Debug.Print "Unable to terminate process: " & proc.hProcess
         Call WriteError(errTerminateProcessFailed, mstrSource, _
               ApiError(GetLastError()))
      Else
         ' Should always come here!
         GetExitCodeProcess proc.hProcess, nCode
         If nCode = STILL_ACTIVE Then
            ' Write an error and close the handles to the
            ' process anyway
            Call WriteError(errTerminateProcessFailed, mstrSource)
         End If
      End If

      ' Close all open handles to the shelled process, even
      ' if any of the above calls error out
      CloseHandle proc.hProcess
      moTimer.Enabled = False
      mfShelling = False
      RaiseEvent Terminated

   End If

   Exit Sub

AbortErr:
   Call LogErrors(Errors)
   mstrSource = mstrModuleName & "Abort"
   On Error GoTo 0
   Err.Raise vbObjectError + errProgramError, _
      mstrSource, _
      LoadResString(errProgramError)

End Sub
Private Sub moTimer_Timer()
   Dim nCode As Long

   GetExitCodeProcess proc.hProcess, nCode
   If nCode <> STILL_ACTIVE Then
      CloseHandle proc.hProcess
      moTimer.Enabled = False
      mfShelling = False
      RaiseEvent Terminated
   End If
End Sub
```

cConnDtl.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConnDtl"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cConnDtl.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of a connection.
'              Contains functions to insert, update and delete
'              connection_dtls records from the database.
```

```
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
Option Base 0

' Local variable(s) to hold property value(s)
Public WorkspaceId As Long
Public ConnNameId As Long
Public ConnName As String
Public ConnectionString As String
Public ConnType As ConnectionType
Public Position As Long
Public NodeDB As Database

Private mintOperation As Operation

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cConnDtl."

' The cSequence class is used to generate unique Connection identifiers
Private mConnectionSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM

Private Sub AssignParameters(qyExec As DAO.QueryDef)
   ' Assigns values to the parameters in the querydef object
   ' The parameter names are cryptic to differentiate them from the field names.
   ' When the parameter names are the same as the field names, parameters in the
where
   ' clause do not get created.

   Dim prmParam As DAO.Parameter

   On Error GoTo AssignParametersErr

   For Each prmParam In qyExec.Parameters
      Select Case prmParam.Name
         Case "[w_id]"
            prmParam.Value = WorkspaceId

         Case "[c_id]"
            prmParam.Value = ConnNameId

         Case "[c_name]"
            prmParam.Value = ConnName

         Case "[c_str]"
            prmParam.Value = ConnectionString

         Case "[c_type]"
            prmParam.Value = ConnType

         Case Else
            ' Write the parameter name that is faulty
            WriteError errInvalidParameter, mstrSource, prmParam.Name
            On Error GoTo 0
            Err.Raise errInvalidParameter, mstrModuleName &
"AssignParameters", _
                  LoadResString(errInvalidParameter)
      End Select
   Next prmParam

   Exit Sub

AssignParametersErr:

   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errAssignParametersFailed, _
      mstrModuleName & "AssignParameters",
LoadResString(errAssignParametersFailed)

End Sub

Public Function Clone() As cConnDtl

   ' Creates a copy of a given Connection

   Dim cCloneConn As cConnDtl

   On Error GoTo CloneErr

   Set cCloneConn = New cConnDtl

   ' Copy all the Connection properties to the newly created Connection
   cCloneConn.WorkspaceId = WorkspaceId
   cCloneConn.ConnNameId = ConnNameId
   cCloneConn.ConnName = ConnName
   cCloneConn.ConnectionString = ConnectionString
   cCloneConn.ConnType = ConnType
   cCloneConn.IndOperation = mintOperation
   cCloneConn.Position = Position

   ' And set the return value to the newly created Connection
   Set Clone = cCloneConn
   Set cCloneConn = Nothing

   Exit Function

CloneErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Clone"
   On Error GoTo 0
   Err.Raise vbObjectError + errCloneFailed, mstrSource,
LoadResString(errCloneFailed)

End Function
Private Sub CheckDupConnectionName()
   ' Check if the Connection name already exists in the workspace

   Dim rstConnection As Recordset
   Dim strSql As String
   Dim qy As DAO.QueryDef

   On Error GoTo CheckDupConnectionNameErr
   mstrSource = mstrModuleName & "CheckDupConnectionName"

   ' Create a recordset object to retrieve the count of all Connections
   ' for the workspace with the same name
   strSql = "Select count(*) as Connection_count " & _
      " from " & TBL_CONNECTION_DTLS & _
      " where " & FLD_ID_WORKSPACE & " = [w_id]" & _
      " and " & FLD_CONN_DTL_CONNECTION_NAME & " = [c_name]" & _
      " and " & FLD_ID_CONN_NAME & " <> [c_id]"

   Set qy = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)
   Call AssignParameters(qy)

   Set rstConnection = qy.OpenRecordset(dbOpenForwardOnly)

   If rstConnection![Connection_count] > 0 Then
      rstConnection.Close
      qy.Close
      ShowError errDupConnDtlName
      On Error GoTo 0
      Err.Raise vbObjectError + errDupConnDtlName, _
         mstrSource, LoadResString(errDupConnDtlName)
   End If

   rstConnection.Close
   qy.Close

   Exit Sub

CheckDupConnectionNameErr:
```

```
    LogErrors Errors
    mstrSource = mstrModuleName & "CheckDupConnectionName"
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, _
        mstrSource, LoadResString(errProgramError)

End Sub
Public Property Let IndOperation(ByVal vdata As Operation)

    ' The valid operations are define in the cOperations
    ' class. Check if the operation is valid
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            BugAssert True
    End Select

End Property
Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    If ConnName = gstrEmptyString Then

        ShowError errConnectionNameMandatory
        On Error GoTo 0
        ' Propogate this error back to the caller
        Err.Raise vbObjectError + errConnectionNameMandatory, _
            mstrSource, LoadResString(errConnectionNameMandatory)
    End If

    ' Raise an error if the Connection name already exists in the workspace
    Call CheckDupConnectionName

End Sub
Public Sub Add()

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddErr

    ' Validate the record before trying to insert the record
    Call Validate

    ' Create a temporary querydef object
    strInsert = "insert into " & TBL_CONNECTION_DTLS & _
            "( " & FLD_ID_WORKSPACE & _
            ", " & FLD_ID_CONN_NAME & _
            ", " & FLD_CONN_DTL_CONNECTION_NAME & _
            ", " & FLD_CONN_DTL_CONNECTION_STRING & _
            ", " & FLD_CONN_DTL_CONNECTION_TYPE & " ) " & _
            " values ( [w_id], [c_id], " & _
            " [c_name], [c_str], [c_type] ) "

    Set qy = dbsAttTool.CreateQueryDef(gstrEmptyString, strInsert)

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

    qy.Execute dbFailOnError
    qy.Close

    Exit Sub

AddErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertFailed, _
        mstrModuleName & "Add", LoadResString(errInsertFailed)

End Sub
Public Sub Delete()

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr

    strDelete = "delete from " & TBL_CONNECTION_DTLS & _
            " where " & FLD_ID_CONN_NAME & " = [c_id]"
    Set qy = dbsAttTool.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteFailed, _
            mstrModuleName & "Delete", LoadResString(errDeleteFailed)

End Sub

Public Sub Modify()

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo ModifyErr

    ' Validate the updated values before trying to modify the db
    Call Validate

    ' Create a temporary querydef object with the modify string
    strUpdate = "update " & TBL_CONNECTION_DTLS & _
            " set " & FLD_ID_WORKSPACE & " = [w_id], " & _
            FLD_CONN_DTL_CONNECTION_NAME & " = [c_name], " & _
            FLD_CONN_DTL_CONNECTION_STRING & " = [c_str], " & _
            FLD_CONN_DTL_CONNECTION_TYPE & " = [c_type] " & _
            " where " & FLD_ID_CONN_NAME & " = [c_id]"
    Set qy = dbsAttTool.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the Connection values to the
    ' querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

ModifyErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyFailed, _
        mstrModuleName & "Modify", LoadResString(errModifyFailed)

End Sub
Public Property Get NextIdentifier() As Long

    Dim lngNextId As Long

    On Error GoTo NextIdentifierErr

    ' Retrieve the next identifier using the sequence class
    Set mConnectionSeq = New cSequence
```

```
    Set mConnectionSeq.IdDatabase = dbsAttTool
    mConnectionSeq.IdentifierColumn = FLD_ID_CONN_NAME
    lngNextId = mConnectionSeq.Identifier
    Set mConnectionSeq = Nothing

    NextIdentifier = lngNextId
    Exit Property

NextIdentifierErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errIdGetFailed, _
        mstrModuleName & "NextIdentifier", LoadResString(errIdGetFailed)

End Property
Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property

Private Sub Class_Initialize()

    Set mFieldValue = New cStringSM

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

    ConnType = giDefaultConnType

End Sub

Private Sub Class_Terminate()

    Set mFieldValue = Nothing

End Sub
```

cConnDtls.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConnDtls"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:      cConnDtls.cls
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cConnDtl objects.
'             Type-safe wrapper around cNodeCollections.
'             Also contains additional functions to determine the connection
'             string value, validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrConnDtls As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cConnDtls."

Public Property Set ConnDb(vdata As Database)

    Set mcarrConnDtls.NodeDB = vdata

End Property
Public Sub Modify(cModifiedConn As cConnDtl)

    ' First check if the parameter record is valid
    Call CheckDupConnName(cModifiedConn)

    Call mcarrConnDtls.Modify(cModifiedConn)

End Sub
Public Sub Load(ByRef cConnToAdd As cConnDtl)

    Call mcarrConnDtls.Load(cConnToAdd)

End Sub
Public Sub Add(ByRef cConnToAdd As cConnDtl)

    ' First check if the record is valid
    Call Validate(cConnToAdd)

    ' Retrieve a unique identifier
    cConnToAdd.ConnNameId = cConnToAdd.NextIdentifier

    Call mcarrConnDtls.Add(cConnToAdd)

End Sub

Public Sub Unload(lConnNameId As Long)

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lConnNameId)

    Call mcarrConnDtls.Unload(lngDeleteElement)

End Sub

Public Sub SaveConnDtlsInWsp(ByVal lngWorkspace As Long)
    ' Call a procedure to save all connection details records for the workspace
    Call mcarrConnDtls.Save(lngWorkspace)

End Sub
Public Function GetConnectionDtl(ByVal lngWorkspace As Long, _
        ByVal strConnectionName As String) As cConnDtl
    ' Returns the connection dtl for the passed in connection name

    Dim lngIndex As Long

    ' Find all parameters in the array with a matching workspace id
    For lngIndex = 0 To mcarrConnDtls.Count - 1
        If mcarrConnDtls(lngIndex).WorkspaceId = lngWorkspace And _
                mcarrConnDtls(lngIndex).ConnName = strConnectionName Then

            Set GetConnectionDtl = mcarrConnDtls(lngIndex)
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcarrConnDtls.Count - 1 Then
        ' The parameter has not been defined for the workspace
        ' Raise an error
        On Error GoTo 0
        Err.Raise vbObjectError + errConnNameInvalid, mstrModuleName &
"GetConnection", _
                LoadResString(errConnNameInvalid)
    End If

End Function
Public Sub Delete(lConnNameId As Long)
    ' Delete the passed in parameter

    Dim lngDeleteElement As Long
```

```
    lngDeleteElement = QueryIndex(lConnNameId)
    Call mcarrConnDtls.Delete(lngDeleteElement)

End Sub
Private Function QueryIndex(lConnNameId As Long) As Long

    Dim lngIndex As Long

    ' Find the matching parameter record in the array
    For lngIndex = 0 To mcarrConnDtls.Count - 1
        If mcarrConnDtls(lngIndex).ConnNameId = lConnNameId And _
                mcarrConnDtls(lngIndex).IndOperation <> DeleteOp Then
            QueryIndex = lngIndex
            Exit Function
        End If
    Next lngIndex

    ' Raise error that parameter has not been found
    On Error GoTo 0
    Err.Raise vbObjectError + errQueryIndexFailed, "cArrParameters.QueryIndex",
_
        LoadResString(errQueryIndexFailed)

End Function

Public Function QueryConnDtl(lConnNameId As Long) As cConnDtl

    Dim lngQueryElement As Long

    lngQueryElement = QueryIndex(lConnNameId)

    ' Return the queried connection object
    Set QueryConnDtl = mcarrConnDtls(lngQueryElement)

End Function
Public Property Get Count() As Long

    Count = mcarrConnDtls.Count

End Property
Public Property Get Item(lngIndex As Long) As cConnDtl
Attribute Item.VB_UserMemId = 0

    Set Item = mcarrConnDtls(lngIndex)

End Property

Private Sub Validate(ByVal cConnToValidate As cConnDtl)
    ' This procedure is necessary since the class cannot validate
    ' all the connection_dtl properties on it's own. This is 'coz we
    ' might have created new connections in the workspace, but not
    ' saved them to the database yet - hence the duplicate check
    ' has to be repeated in the array

    Dim lngIndex As Long
    Dim cTempParam As cConnDtl

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrConnDtls.Count - 1
        Set cTempParam = mcarrConnDtls(lngIndex)
        If cTempParam.WorkspaceId = cConnToValidate.WorkspaceId And _
                cTempParam.ConnName = cConnToValidate.ConnName And _
                cTempParam.IndOperation <> DeleteOp Then
            On Error GoTo 0
            Err.Raise vbObjectError + errDupConnDtlName, _
                mstrSource, LoadResString(errDupConnDtlName)
        End If
    Next lngIndex

End Sub
Private Sub CheckDupConnName(ByVal cConnToValidate As cConnDtl)

    Dim lngIndex As Long
    Dim cTempParam As cConnDtl

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrConnDtls.Count - 1
        Set cTempParam = mcarrConnDtls(lngIndex)
        If cTempParam.WorkspaceId = cConnToValidate.WorkspaceId And _
                cTempParam.ConnName = cConnToValidate.ConnName And _
                cTempParam.ConnNameId <> cConnToValidate.ConnNameId And _
                cTempParam.IndOperation <> DeleteOp Then
            ShowError errDupConnDtlName
            On Error GoTo 0
            Err.Raise vbObjectError + errDupConnDtlName, _
                mstrSource, LoadResString(errDupConnDtlName)
        End If
    Next lngIndex

End Sub

Private Sub Class_Initialize()

    Set mcarrConnDtls = New cNodeCollections

End Sub

Private Sub Class_Terminate()

    Set mcarrConnDtls = Nothing

End Sub
```

cConnection.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConnection"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cConnection.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of a connection string.
'              Contains functions to insert, update and delete
'              workspace_connections records from the database.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
Option Base 0

' Local variable(s) to hold property value(s)
Public WorkspaceId As Long
Public ConnectionId As Long
Public ConnectionValue As String
Public Description As String
Public NodeDB As Database
Public Position As Long
Public NoCountDisplay As Boolean
Public NoExecute As Boolean
Public ParseQueryOnly As Boolean
Public QuotedIdentifiers As Boolean
Public AnsiNulls As Boolean
Public ShowQueryPlan As Boolean
Public ShowStatsTime As Boolean
Public ShowStatsIO As Boolean
Public ParseOdbcMsg As Boolean
Public RowCount As Long
```

```
Public TsqlBatchSeparator As String
Public QueryTimeOut As Long
Public ServerLanguage As String
Public CharacterTranslation As Boolean
Public RegionalSettings As Boolean

Private mstrConnectionName As String
Private mintOperation As Operation

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cConnection."

' The cSequence class is used to generate unique Connection identifiers
Private mConnectionSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM

Private Sub AssignParameters(qyExec As DAO.QueryDef)
   ' Assigns values to the parameters in the querydef object
   ' The parameter names are cryptic to differentiate them from the field names.
   ' When the parameter names are the same as the field names, parameters in the
where
   ' clause do not get created.

   Dim prmParam As DAO.Parameter

   On Error GoTo AssignParametersErr

   For Each prmParam In qyExec.Parameters
      Select Case prmParam.Name
         Case "[w_id]"
            prmParam.Value = WorkspaceId

         Case "[c_id]"
            prmParam.Value = ConnectionId

         Case "[c_name]"
            prmParam.Value = mstrConnectionName

         Case "[c_value]"
            prmParam.Value = ConnectionValue

         Case "[desc]"
            prmParam.Value = Description

         Case "[no_count]"
            prmParam.Value = NoCountDisplay

         Case "[no_exec]"
            prmParam.Value = NoExecute

         Case "[parse_only]"
            prmParam.Value = ParseQueryOnly

         Case "[quoted_id]"
            prmParam.Value = QuotedIdentifiers

         Case "[a_nulls]"
            prmParam.Value = AnsiNulls

         Case "[show_qp]"
            prmParam.Value = ShowQueryPlan

         Case "[stats_tm]"
            prmParam.Value = ShowStatsTime

         Case "[stats_io]"
            prmParam.Value = ShowStatsIO

         Case "[parse_odbc]"
            prmParam.Value = ParseOdbcMsg

         Case "[row_cnt]"
            prmParam.Value = RowCount

         Case "[batch_sep]"
            prmParam.Value = TsqlBatchSeparator

         Case "[qry_tmout]"
            prmParam.Value = QueryTimeOut

         Case "[lang]"
            prmParam.Value = ServerLanguage

         Case "[char_trans]"
            prmParam.Value = CharacterTranslation

         Case "[reg_settings]"
            prmParam.Value = RegionalSettings

         Case Else
            ' Write the parameter name that is faulty
            WriteError errInvalidParameter, mstrSource, prmParam.Name
            On Error GoTo 0
            Err.Raise errInvalidParameter, mstrModuleName &
"AssignParameters", _
                  LoadResString(errInvalidParameter)
      End Select
   Next prmParam

   Exit Sub

AssignParametersErr:

   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errAssignParametersFailed, _
      mstrModuleName & "AssignParameters",
LoadResString(errAssignParametersFailed)

End Sub

Public Function Clone() As cConnection

   ' Creates a copy of a given Connection

   Dim cCloneConn As cConnection

   On Error GoTo CloneErr

   Set cCloneConn = New cConnection

   ' Copy all the Connection properties to the newly
   ' created Connection
   Set cCloneConn.NodeDB = NodeDB
   cCloneConn.WorkspaceId = WorkspaceId
   cCloneConn.ConnectionId = ConnectionId
   cCloneConn.ConnectionName = mstrConnectionName
   cCloneConn.ConnectionValue = ConnectionValue
   cCloneConn.Description = Description
   cCloneConn.IndOperation = mintOperation
   cCloneConn.Position = Position
   cCloneConn.NoCountDisplay = NoCountDisplay
   cCloneConn.NoExecute = NoExecute
   cCloneConn.ParseQueryOnly = ParseQueryOnly
   cCloneConn.QuotedIdentifiers = QuotedIdentifiers
   cCloneConn.AnsiNulls = AnsiNulls
   cCloneConn.ShowQueryPlan = ShowQueryPlan
   cCloneConn.ShowStatsTime = ShowStatsTime
   cCloneConn.ShowStatsIO = ShowStatsIO
   cCloneConn.ParseOdbcMsg = ParseOdbcMsg
   cCloneConn.RowCount = RowCount
   cCloneConn.TsqlBatchSeparator = TsqlBatchSeparator
   cCloneConn.QueryTimeOut = QueryTimeOut
   cCloneConn.ServerLanguage = ServerLanguage
```

```
    cCloneConn.CharacterTranslation = CharacterTranslation
    cCloneConn.RegionalSettings = RegionalSettings

    ' And set the return value to the newly created Connection
    Set Clone = cCloneConn
    Set cCloneConn = Nothing

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, mstrSource,
LoadResString(errCloneFailed)

End Function
Private Sub CheckDupConnectionName()
    ' Check if the Connection name already exists in the workspace

    Dim rstConnection As Recordset
    Dim strSql As String
    Dim qy As DAO.QueryDef

    On Error GoTo CheckDupConnectionNameErr
    mstrSource = mstrModuleName & "CheckDupConnectionName"

    ' Create a recordset object to retrieve the count of all Connections
    ' for the workspace with the same name
    strSql = "Select count(*) as Connection_count " & _
        " from workspace_connections " & _
        " where workspace_id = [w_id]" & _
        " and connection_name = [c_name]" & _
        " and connection_id <> [c_id]"

    Set qy = NodeDB.CreateQueryDef(gstrEmptyString, strSql)
    Call AssignParameters(qy)

    Set rstConnection = qy.OpenRecordset(dbOpenForwardOnly)

    If rstConnection![Connection_count] > 0 Then
        rstConnection.Close
        qy.Close
        ShowError errDuplicateConnectionName
        On Error GoTo 0
        Err.Raise vbObjectError + errDuplicateConnectionName, _
            mstrSource, LoadResString(errDuplicateConnectionName)
    End If

    rstConnection.Close
    qy.Close

    Exit Sub

CheckDupConnectionNameErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "CheckDupConnectionName"
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, _
        mstrSource, LoadResString(errProgramError)

End Sub
Private Sub CheckDB()
    ' Check if the database object has been initialized

    If NodeDB Is Nothing Then
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidDB, _
            mstrModuleName & "CheckDB", LoadResString(errInvalidDB)
    End If

End Sub
Public Property Let ConnectionName(vdata As String)

    If vdata = gstrEmptyString Then

        ShowError errConnectionNameMandatory
        On Error GoTo 0
        ' Propogate this error back to the caller
        Err.Raise vbObjectError + errConnectionNameMandatory, _
            mstrSource, LoadResString(errConnectionNameMandatory)
    Else
        mstrConnectionName = vdata
    End If

End Property

Public Property Let IndOperation(ByVal vdata As Operation)

    ' The valid operations are define in the cOperations
    ' class. Check if the operation is valid
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            BugAssert True
    End Select

End Property
Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    ' Check if the db object is valid
    Call CheckDB

    ' Raise an error if the Connection name already exists in the workspace
    Call CheckDupConnectionName

End Sub
Public Sub Add()

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddErr

    ' Validate the record before trying to insert the record
    Call Validate

    ' Create a temporary querydef object
    strInsert = "insert into workspace_connections " & _
            "( workspace_id, connection_id, " & _
            "connection_name, connection_value, " & _
            "description, no_count_display, " & _
            "no_execute, parse_query_only, " & _
            "ANSI_quoted_identifiers, ANSI_nulls, " & _
            "show_query_plan, show_stats_time, " & _
            "show_stats_io, parse_odbc_msg_prefixes, " & _
            "row_count, tsql_batch_separator, " & _
            "query_time_out, server_language, " & _
            "character_translation, regional_settings ) " & _
           " values ( [w_id], [c_id], [c_name], [c_value], " & _
           " [desc], [no_count], [no_exec], [parse_only], " & _
           " [quoted_id], [a_nulls], [show_qp], [stats_tm], " & _
           " [stats_io], [parse_odbc], [row_cnt], [batch_sep], " & _
           " [qry_tmout], [lang], [char_trans], [reg_settings] ) "

    Set qy = NodeDB.CreateQueryDef(gstrEmptyString, strInsert)

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

    qy.Execute dbFailOnError
```

```
    qy.Close

    Exit Sub

AddErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertFailed, _
        mstrModuleName & "Add", LoadResString(errInsertFailed)

End Sub
Public Sub Delete()

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr

    ' Check if the db object is valid
    Call CheckDB

    strDelete = "delete from workspace_connections " & _
            " where connection_id = [c_id]"
    Set qy = NodeDB.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteFailed, _
            mstrModuleName & "Delete", LoadResString(errDeleteFailed)

End Sub

Public Sub Modify()

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo ModifyErr

    ' Validate the updated values before trying to modify the db
    Call Validate

    ' Create a temporary querydef object with the modify string
    strUpdate = "update workspace_connections " & _
            " set workspace_id = [w_id], " & _
            "connection_name = [c_name], " & _
            "connection_value = [c_value], " & _
            "description = [desc], " & _
            "no_count_display = [no_count], " & _
            "no_execute = [no_exec], " & _
            "parse_query_only = [parse_only], " & _
            "ANSI_quoted_identifiers = [quoted_id], " & _
            "ANSI_nulls = [a_nulls], " & _
            "show_query_plan = [show_qp], " & _
            "show_stats_time = [stats_tm], " & _
            "show_stats_io = [stats_io], " & _
            "parse_odbc_msg_prefixes = [parse_odbc], " & _
            "row_count = [row_cnt], " & _
            "tsql_batch_separator = [batch_sep], " & _
            "query_time_out = [qry_tmout], " & _
            "server_language = [lang], " & _
            "character_translation = [char_trans], " & _
            "regional_settings = [reg_settings] " & _
            " where connection_id = [c_id]"
    Set qy = NodeDB.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the Connection values to the
    ' querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

ModifyErr:

    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyFailed, _
        mstrModuleName & "Modify", LoadResString(errModifyFailed)

End Sub
Public Property Get ConnectionName() As String

    ConnectionName = mstrConnectionName

End Property

Public Property Get NextIdentifier() As Long

    Dim lngNextId As Long

    On Error GoTo NextIdentifierErr

    ' First check if the database object is valid
    Call CheckDB

    ' Retrieve the next identifier using the sequence class
    Set mConnectionSeq = New cSequence
    Set mConnectionSeq.IdDatabase = NodeDB
    mConnectionSeq.IdentifierColumn = "connection_id"
    lngNextId = mConnectionSeq.Identifier
    Set mConnectionSeq = Nothing

    NextIdentifier = lngNextId
    Exit Property

NextIdentifierErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errIdGetFailed, _
        mstrModuleName & "NextIdentifier", LoadResString(errIdGetFailed)

End Property
Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property

Private Sub Class_Initialize()

    Set mFieldValue = New cStringSM

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

    ' Initialize connection properties to their default values
    NoCountDisplay = DEF_NO_COUNT_DISPLAY
    NoExecute = DEF_NO_EXECUTE
    ParseQueryOnly = DEF_PARSE_QUERY_ONLY
    QuotedIdentifiers = DEF_ANSI_QUOTED_IDENTIFIERS
    AnsiNulls = DEF_ANSI_NULLS
    ShowQueryPlan = DEF_SHOW_QUERY_PLAN
    ShowStatsTime = DEF_SHOW_STATS_TIME
    ShowStatsIO = DEF_SHOW_STATS_IO
```

```
    ParseOdbcMsg = DEF_PARSE_ODBC_MSG_PREFIXES
    RowCount = DEF_ROW_COUNT
    TsqlBatchSeparator = DEF_TSQL_BATCH_SEPARATOR
    QueryTimeOut = DEF_QUERY_TIME_OUT
    ServerLanguage = DEF_SERVER_LANGUAGE
    CharacterTranslation = DEF_CHARACTER_TRANSLATION
    RegionalSettings = DEF_REGIONAL_SETTINGS

End Sub

Private Sub Class_Terminate()

    Set NodeDB = Nothing
    Set mFieldValue = Nothing

End Sub
```

cConnections.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConnections"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:      cConnections.cls
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Implements an array of cConnection objects.
'             Type-safe wrapper around cNodeCollections.
'             Also contains validation functions, etc.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcarrConnections As cNodeCollections

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cConnections."

Public Property Set ConnDb(vdata As Database)

    Set mcarrConnections.NodeDB = vdata

End Property
Public Sub Modify(cModifiedConn As cConnection)

    ' First check if the parameter record is valid
    Call CheckDupConnName(cModifiedConn)

    Call mcarrConnections.Modify(cModifiedConn)

End Sub
Public Sub Load(ByRef cConnToAdd As cConnection)

    Call mcarrConnections.Load(cConnToAdd)

End Sub
Public Sub Add(ByRef cConnToAdd As cConnection)

    Set cConnToAdd.NodeDB = mcarrConnections.NodeDB

    ' First check if the record is valid
    Call Validate(cConnToAdd)

    ' Retrieve a unique identifier
    cConnToAdd.ConnectionId = cConnToAdd.NextIdentifier

    Call mcarrConnections.Add(cConnToAdd)

End Sub

Public Sub Unload(lngConnId As Long)

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lngConnId)

    Call mcarrConnections.Unload(lngDeleteElement)

End Sub

Public Sub SaveConnectionsInWsp(ByVal lngWorkspace As Long)
    ' Call a procedure to save all connection records for the workspace
    Call mcarrConnections.Save(lngWorkspace)

End Sub
Public Function GetConnection(ByVal lngWorkspace As Long, _
        ByVal strConnectionName As String) As cConnection
    ' Returns the connection string for the passed in connection name

    Dim lngIndex As Long

    ' Find all parameters in the array with a matching workspace id
    For lngIndex = 0 To mcarrConnections.Count - 1
        If mcarrConnections(lngIndex).WorkspaceId = lngWorkspace And _
            mcarrConnections(lngIndex).ConnectionName = strConnectionName
Then

            Set GetConnection = mcarrConnections(lngIndex)
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcarrConnections.Count - 1 Then
        ' The parameter has not been defined for the workspace
        ' Raise an error
        On Error GoTo 0
        Err.Raise vbObjectError + errConnNameInvalid, mstrModuleName &
"GetConnection", _
            LoadResString(errConnNameInvalid)
    End If

End Function
Public Sub Delete(lngConnId As Long)
    ' Delete the passed in parameter

    Dim lngDeleteElement As Long

    lngDeleteElement = QueryIndex(lngConnId)
    Call mcarrConnections.Delete(lngDeleteElement)

End Sub
Private Function QueryIndex(lngConnId As Long) As Long

    Dim lngIndex As Long

    ' Find the matching parameter record in the array
    For lngIndex = 0 To mcarrConnections.Count - 1
        If mcarrConnections(lngIndex).ConnectionId = lngConnId And _
            mcarrConnections(lngIndex).IndOperation <> DeleteOp Then
            QueryIndex = lngIndex
            Exit Function
        End If
    Next lngIndex

    ' Raise error that parameter has not been found
    On Error GoTo 0
```

```
    Err.Raise vbObjectError + errQueryIndexFailed, "cArrParameters.QueryIndex",
_
        LoadResString(errQueryIndexFailed)

End Function

Public Function QueryConnection(lngConnId As Long) As cConnection

    Dim lngQueryElement As Long

    lngQueryElement = QueryIndex(lngConnId)

    ' Return the queried connection object
    Set QueryConnection = mcarrConnections(lngQueryElement)

End Function
Public Property Get Count() As Long

    Count = mcarrConnections.Count

End Property
Public Property Get Item(lngIndex As Long) As cConnection
Attribute Item.VB_UserMemId = 0

    Set Item = mcarrConnections(lngIndex)

End Property

Public Sub Validate(ByVal cConnToValidate As cConnection)
    ' This procedure is necessary since the class cannot validate
    ' all the parameter properties on it's own. This is 'coz we
    ' might have created new parameters in the workspace, but not
    ' saved them to the database yet - hence the duplicate check
    ' has to be repeated in the array

    Dim lngIndex As Long
    Dim cTempParam As cConnection

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrConnections.Count - 1
        Set cTempParam = mcarrConnections(lngIndex)
        If cTempParam.WorkspaceId = cConnToValidate.WorkspaceId And _
            cTempParam.ConnectionName = cConnToValidate.ConnectionName
And _
            cTempParam.IndOperation <> DeleteOp Then
            On Error GoTo 0
            Err.Raise vbObjectError + errDuplicateConnectionName, _
                mstrSource, LoadResString(errDuplicateConnectionName)
        End If
    Next lngIndex

End Sub
Public Sub CheckDupConnName(ByVal cConnToValidate As cConnection)

    Dim lngIndex As Long
    Dim cTempParam As cConnection

    ' Check if the parameter name already exists in the workspace
    For lngIndex = 0 To mcarrConnections.Count - 1
        Set cTempParam = mcarrConnections(lngIndex)
        If cTempParam.WorkspaceId = cConnToValidate.WorkspaceId And _
            cTempParam.ConnectionName = cConnToValidate.ConnectionName
And _
            cTempParam.ConnectionId <> cConnToValidate.ConnectionId And _
            cTempParam.IndOperation <> DeleteOp Then
            ShowError errDuplicateConnectionName
            On Error GoTo 0
            Err.Raise vbObjectError + errDuplicateConnectionName, _
                mstrSource, LoadResString(errDuplicateConnectionName)
        End If
    Next lngIndex

End Sub
Private Sub Class_Initialize()

    Set mcarrConnections = New cNodeCollections

End Sub

Private Sub Class_Terminate()

    Set mcarrConnections = Nothing

End Sub
```

cConstraint.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cConstraint"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cConstraint.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of a constraint.
'              Contains functions to insert, update and delete
'              step_constraints records from the database.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Module level variables to store the property values
Private mlngConstraintId As Long
Private mlngStepId As Long
Private mstrVersionNo As String
Private mintConstraintType As Integer
Private mlngGlobalStepId As Long
Private mstrGlobalVersionNo As String
Private mintSequenceNo As Integer
Private mdbsConstraintDB As Database
Private mlngWorkspaceId As Integer
Private mintOperation As Operation
Private mlngPosition As Long

' The cSequence class is used to generate unique step identifiers
Private mConstraintSeq As cSequence

Private Const mstrModuleName As String = ".cConstraint."
Private mstrSource As String

Public Enum ConstraintType
    gintPreStep = 1
    gintPostStep = 2
End Enum

Private Const mstrSQ As String = """
Public Property Get WorkspaceId() As Long
    WorkspaceId = mlngWorkspaceId
End Property
Public Property Let WorkspaceId(ByVal vdata As Long)
    mlngWorkspaceId = vdata
End Property

Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property
```

```
Public Property Let IndOperation(ByVal vdata As Operation)

   On Error GoTo IndOperationErr
   mstrSource = mstrModuleName & "IndOperation"

   ' The valid operations are define in the cOperations
   ' class. Check if the operation is valid
   Select Case vdata
      Case QueryOp, InsertOp, UpdateOp, DeleteOp
         mintOperation = vdata

      Case Else
         On Error GoTo 0
         Err.Raise vbObjectError + errInvalidOperation, _
               mstrSource, LoadResString(errInvalidOperation)
   End Select

   Exit Property

IndOperationErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "IndOperation"
   On Error GoTo 0
   Err.Raise vbObjectError + errLetOperationFailed, _
      mstrSource, LoadResString(errLetOperationFailed)

End Property

Public Function Clone() As cConstraint

   ' Creates a copy of a given constraint

   Dim cConsClone As cConstraint

   On Error GoTo CloneErr
   mstrSource = mstrModuleName & "Clone"

   Set cConsClone = New cConstraint

   ' Copy all the workspace properties to the newly
   ' created workspace
   cConsClone.ConstraintId = mlngConstraintId
   cConsClone.StepId = mlngStepId
   cConsClone.VersionNo = mstrVersionNo
   cConsClone.ConstraintType = mintConstraintType
   cConsClone.GlobalStepId = mlngGlobalStepId
   cConsClone.GlobalVersionNo = mstrGlobalVersionNo
   cConsClone.SequenceNo = mintSequenceNo
   cConsClone.WorkspaceId = mlngWorkspaceId
   cConsClone.IndOperation = mintOperation

   ' And set the return value to the newly created constraint
   Set Clone = cConsClone

   Exit Function

CloneErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Clone"
   On Error GoTo 0
   Err.Raise vbObjectError + errCloneFailed, _
         mstrSource, LoadResString(errCloneFailed)

End Function

Public Property Get SequenceNo() As Integer

   SequenceNo = mintSequenceNo

End Property

Public Property Let SequenceNo(ByVal vdata As Integer)
   mintSequenceNo = vdata
End Property

Public Sub Add()
   ' Inserts a new step constraint into the database

   Dim strInsert As String
   Dim qy As DAO.QueryDef

   On Error GoTo AddErr

   ' First check if the database object is valid
   Call CheckDB

   ' Any record validations
   Call Validate

   ' Create a temporary querydef object
   strInsert = "insert into step_constraints " & _
         "( constraint_id, step_id, version_no, " & _
         " constraint_type, global_step_id, global_version_no, sequence_no ) " & _
         " values ( [cons_id], [s_id], [ver_no], " & _
         " [cons_type], [g_step_id], [g_ver_no], " & _
         " [seq_no] )"
   Set qy = mdbsConstraintDB.CreateQueryDef(gstrEmptyString, strInsert)

   ' Call a procedure to execute the Querydef object
   Call AssignParameters(qy)

   qy.Execute dbFailOnError
   qy.Close

'   strInsert = "insert into step_constraints " & _
'      "( constraint_id, step_id, version_no, " & _
'      " constraint_type, global_step_id, global_version_no, sequence_no ) " & _
'      " values ( " & _
'      Str(mlngConstraintId) & ", " & Str(mlngStepId) & ", " & _
'      mstrSQ & mstrVersionNo & mstrSQ & ", " & Str(mintConstraintType) & ",
" & _
'      Str(mlngGlobalStepId) & ", " & mstrSQ & mstrGlobalVersionNo & mstrSQ
& ", " & _
'      Str(mintSequenceNo) & " ) "
'
'   BugMessage strInsert
'   mdbsConstraintDB.Execute strInsert, dbFailOnError
   Exit Sub

AddErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Add"
   On Error GoTo 0
   Err.Raise vbObjectError + errAddConstraintFailed, _
         mstrSource, _
         LoadResString(errAddConstraintFailed)
End Sub
Private Sub AssignParameters(qyExec As DAO.QueryDef)
   ' Assigns values to the parameters in the querydef object
   ' The parameter names are cryptic to make them different
   ' from the field names. When the parameter names are
   ' the same as the field names, parameters in the where
   ' clause do not get created.

   Dim prmParam As DAO.Parameter

   On Error GoTo AssignParametersErr
   mstrSource = mstrModuleName & "AssignParameters"

   For Each prmParam In qyExec.Parameters
      Select Case prmParam.Name
         Case "[cons_id]"
            prmParam.Value = mlngConstraintId

         Case "[s_id]"
            prmParam.Value = mlngStepId

         Case "[ver_no]"
```

```
            prmParam.Value = mstrVersionNo

        Case "[cons_type]"
            prmParam.Value = mintConstraintType

        Case "[g_step_id]"
            prmParam.Value = mlngGlobalStepId

        Case "[g_ver_no]"
            prmParam.Value = mstrGlobalVersionNo

        Case "[seq_no]"
            prmParam.Value = mintSequenceNo

        Case Else
            ' Write the parameter name that is faulty
            WriteError errInvalidParameter, mstrSource, _
                prmParam.Name
            On Error GoTo 0
            Err.Raise errInvalidParameter, mstrSource, _
                LoadResString(errInvalidParameter)
    End Select
  Next prmParam

  Exit Sub

AssignParametersErr:

  mstrSource = mstrModuleName & "AssignParameters"
  Call LogErrors(Errors)
  On Error GoTo 0
  Err.Raise vbObjectError + errAssignParametersFailed, _
    mstrSource, LoadResString(errAssignParametersFailed)

End Sub
Public Property Get NextIdentifier() As Long

  Dim lngNextId As Long

  On Error GoTo NextIdentifierErr

  ' First check if the database object is valid
  Call CheckDB

  ' Retrieve the next constraint identifier using the
  ' sequence class
  Set mConstraintSeq = New cSequence
  Set mConstraintSeq.IdDatabase = mdbsConstraintDB
  mConstraintSeq.IdentifierColumn = "constraint_id"
  lngNextId = mConstraintSeq.Identifier
  Set mConstraintSeq = Nothing

  NextIdentifier = lngNextId
  Exit Property

NextIdentifierErr:
  LogErrors Errors
  mstrSource = mstrModuleName & "NextIdentifier"
  On Error GoTo 0
  Err.Raise vbObjectError + errStepIdGetFailed, _
    mstrSource, LoadResString(errStepIdGetFailed)

End Property

Private Sub CheckDB()
  ' Check if the database object has been initialized

  If mdbsConstraintDB Is Nothing Then
    ShowError errInvalidDB
    On Error GoTo 0
    Err.Raise vbObjectError + errInvalidDB, _
      mstrModuleName, LoadResString(errInvalidDB)
  End If
End Sub

Public Sub Delete()
  ' Deletes the step constraint record from the database

  Dim strDelete As String
  Dim qy As DAO.QueryDef

  On Error GoTo DeleteErr
  mstrSource = mstrModuleName & "Delete"

  ' There can be multiple constraints for a step,
  ' meaning that there can be multiple constraint records
  ' with the same constraint_id. Only a combination
  ' of the step_id, version and constraint_id will be
  ' unique
  strDelete = "delete from step_constraints " & _
        " where constraint_id = [cons_id]" & _
        " and step_id = [s_id] " & _
        " and version_no = [ver_no] "
  Set qy = mdbsConstraintDB.CreateQueryDef(gstrEmptyString, strDelete)

  Call AssignParameters(qy)
  qy.Execute dbFailOnError

  qy.Close

'   strDelete = "Delete from step_constraints " & _
'         " where constraint_id = " & Str(mlngConstraintId) & _
'         " and step_id = " & Str(mlngStepId) & _
'         " and version_no = " & mstrSQ & mstrVersionNo & mstrSQ
'
'   'BugMessage strDelete
'   mdbsConstraintDB.Execute strDelete, dbFailOnError

  Exit Sub

DeleteErr:
  LogErrors Errors
  mstrSource = mstrModuleName & "Delete"
  On Error GoTo 0
  Err.Raise vbObjectError + errDeleteConstraintFailed, _
        mstrSource, _
        LoadResString(errDeleteConstraintFailed)
End Sub
Public Sub Modify()
  ' Updates the sequence no of the step constraint record
  ' in the database

  Dim strUpdate As String
  Dim qy As QueryDef

  On Error GoTo Modify

  ' First check if the database object is valid
  Call CheckDB

  ' Any record validations
  Call Validate

  ' There can be multiple constraints for a step,
  ' meaning that there can be multiple constraint records
  ' with the same constraint_id. Only a combination
  ' of the step_id, version and constraint_id will be
  ' unique
  ' Create a temporary querydef object with the modify string
  strUpdate = "Update step_constraints " & _
        " set sequence_no = [seq_no] " & _
        " where constraint_id = [cons_id] " & _
        " and step_id = [s_id] " & _
        " and version_no = [ver_no] "
  Set qy = mdbsConstraintDB.CreateQueryDef(gstrEmptyString, strUpdate)

  ' Call a procedure to assign the parameter values to the
```

```
    ' querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

'    strUpdate = "Update step_constraints " & _
'            " set sequence_no = " & Str(mintSequenceNo) & _
'            " where constraint_id = " & Str(mlngConstraintId) & _
'            " and step_id = " & Str(mlngStepId) & _
'            " and version_no = " & mstrSQ & mstrVersionNo & mstrSQ
'
'    'BugMessage strUpdate
'    mdbsConstraintDB.Execute strUpdate, dbFailOnError
    Exit Sub

Modify:
    LogErrors Errors
    mstrSource = mstrModuleName & "Modify"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateConstraintFailed, _
          mstrSource, _
          LoadResString(errUpdateConstraintFailed)
End Sub
Public Property Get Position() As Long

    Position = mlngPosition

End Property
Public Property Let Position(ByVal RHS As Long)

    mlngPosition = RHS

End Property

Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    ' No validations are necessary for the constraint object

End Sub

Public Property Set NodeDB(vdata As Database)

    Set mdbsConstraintDB = vdata

End Property

Public Property Get NodeDB() As Database

    Set NodeDB = mdbsConstraintDB

End Property

Public Property Get GlobalVersionNo() As String

    GlobalVersionNo = mstrGlobalVersionNo

End Property

Public Property Let GlobalVersionNo(ByVal vdata As String)

    mstrGlobalVersionNo = vdata

End Property

Public Property Get GlobalStepId() As Long

    GlobalStepId = mlngGlobalStepId

End Property

Public Property Get ConstraintId() As Long

    ConstraintId = mlngConstraintId

End Property

Public Property Get VersionNo() As String

    VersionNo = mstrVersionNo

End Property

Public Property Get StepId() As Long

    StepId = mlngStepId

End Property

Public Property Let VersionNo(ByVal vdata As String)

    mstrVersionNo = vdata

End Property

Public Property Let StepId(ByVal vdata As Long)

    mlngStepId = vdata

End Property

Public Property Let ConstraintId(ByVal vdata As Long)
    On Error GoTo ConstraintIdErr
    mstrSource = mstrModuleName & "ConstraintId"

    If (vdata > 0) Then
        mlngConstraintId = vdata
    Else
        ' Propogate this error back to the caller
        On Error GoTo 0
        Err.Raise vbObjectError + errConstraintIdInvalid, _
            mstrSource, LoadResString(errConstraintIdInvalid)
    End If

    Exit Property

ConstraintIdErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "ConstraintId"
    On Error GoTo 0
    Err.Raise vbObjectError + errConstraintIdSetFailed, _
        mstrSource, LoadResString(errConstraintIdSetFailed)

End Property

Public Property Let GlobalStepId(ByVal vdata As Long)

    On Error GoTo GlobalStepIdErr
    mstrSource = mstrModuleName & "GlobalStepId"

    If (vdata > 0) Then
        mlngGlobalStepId = vdata
    Else
        ' Propogate this error back to the caller
        On Error GoTo 0
        Err.Raise vbObjectError + errGlobalStepIdInvalid, _
            mstrSource, LoadResString(errGlobalStepIdInvalid)
    End If

    Exit Property

GlobalStepIdErr:
    LogErrors Errors
```

```
    mstrSource = mstrModuleName & "GlobalStepId"
    On Error GoTo 0
    Err.Raise vbObjectError + errGlobalStepIdSetFailed, _
        mstrSource, LoadResString(errGlobalStepIdSetFailed)

End Property

Public Property Let ConstraintType(ByVal vdata As ConstraintType)

    On Error GoTo ConstraintTypeErr

    ' A global step can be either a pre- or a post-execution step.
    ' These constants have been defined in the enumeration,
    ' ConstraintType, which is exposed
    Select Case vdata
        Case gintPreStep, gintPostStep
            mintConstraintType = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errConstraintTypeInvalid, _
                mstrSource, LoadResString(errConstraintTypeInvalid)
    End Select

    Exit Property

ConstraintTypeErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "ConstraintType"
    On Error GoTo 0
    Err.Raise vbObjectError + errConstraintTypeLetFailed, _
        mstrSource, LoadResString(errConstraintTypeLetFailed)

End Property

Public Property Get ConstraintType() As ConstraintType

    ConstraintType = mintConstraintType

End Property

Private Sub Class_Initialize()

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

End Sub
```

cFailedStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cFailedStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cFailedStep.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     Properties of a step execution failure.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public InstanceId As Long
Public StepId As Long
Public ParentStepId As Long
Public ContCriteria As ContinuationCriteria
Public EndTime As Currency
Public AskResponse As Long
```

cFailedSteps.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cFailedSteps"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cFailedSteps.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     This module encapsulates a collection of failed steps. It
'               also determines whether sub-steps of a passed in step need
'               to be skipped due to a failure.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcFailedSteps As cVector
Public Function ExecuteSubStep(lParentStepId As Long) As Boolean
    ' Returns False if there is any condition that prevents sub-steps of the passed
    ' in instance from being executed
    Dim lIndex As Long

    ExecuteSubStep = True

    For lIndex = 0 To Count() - 1
        If mcFailedSteps(lIndex).ContCriteria = gintOnFailureCompleteSiblings
And _
                lParentStepId <> mcFailedSteps(lIndex).ParentStepId Then
            ExecuteSubStep = False
            Exit For
        End If

        If mcFailedSteps(lIndex).ContCriteria = gintOnFailureAbortSiblings And _
                lParentStepId = mcFailedSteps(lIndex).ParentStepId Then
            ExecuteSubStep = False
            Exit For
        End If

        If mcFailedSteps(lIndex).ContCriteria = gintOnFailureSkipSiblings And _
                lParentStepId = mcFailedSteps(lIndex).ParentStepId Then
            ExecuteSubStep = False
            Exit For
        End If

        If mcFailedSteps(lIndex).ContCriteria = gintOnFailureAbort Then
            ExecuteSubStep = False
            Exit For
        End If

    Next lIndex

End Function
Public Sub Add(ByVal objItem As cFailedStep)

    mcFailedSteps.Add objItem

End Sub
Public Function Delete(ByVal lPosition As Long) As cFailedStep
```

```
    Set Delete = mcFailedSteps.Delete(lPosition)

End Function

Public Sub Clear()

    mcFailedSteps.Clear

End Sub
Public Function Count() As Long

    Count = mcFailedSteps.Count

End Function
Public Property Get Item(ByVal Position As Long) As cFailedStep
Attribute Item.VB_UserMemId = 0

    Set Item = mcFailedSteps.Item(Position)

End Property

Public Function StepFailed(lStepId As Long) As Boolean

    ' Returns True if a failure record already exists for the passed in step
    Dim lIndex As Long

    StepFailed = False

    For lIndex = 0 To Count() - 1
        If mcFailedSteps(lIndex).StepId = lStepId Then
            StepFailed = True
            Exit For
        End If
    Next lIndex

End Function

Private Sub Class_Initialize()

    Set mcFailedSteps = New cVector

End Sub

Private Sub Class_Terminate()

    Set mcFailedSteps = Nothing

End Sub
```

cFileInfo.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cFileInfo"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cFileInfo.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     File Properties viz. name, handle, etc.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mstrFileName As String
Private mintFileHandle As Integer
Private mdbsNodeDb As Database ' Since it is used to form a cNodeCollection
Private mlngPosition As Long ' Since it is used to form a cNodeCollection
Public Property Get FileName() As String

    FileName = mstrFileName

End Property
Public Property Let FileName(ByVal vdata As String)

    mstrFileName = vdata

End Property
Public Property Let FileHandle(ByVal vdata As Integer)

    mintFileHandle = vdata

End Property
Public Property Set NodeDB(vdata As Database)

    Set mdbsNodeDb = vdata

End Property

Public Property Get NodeDB() As Database

    Set NodeDB = mdbsNodeDb

End Property
Public Property Get Position() As Long

    Position = mlngPosition

End Property
Public Property Let Position(ByVal vdata As Long)

    mlngPosition = vdata

End Property

Public Property Get FileHandle() As Integer

    FileHandle = mintFileHandle

End Property
```

cFileSM.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cFileSM"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Attribute VB_Ext_KEY = "SavedWithClassBuilder" ,"Yes"
Attribute VB_Ext_KEY = "Top_Level" ,"Yes"
'  FILE:       cFileSM.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     Encapsulates functions to open a file and write to it.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cFileSM."
```

```
Private mstrSource As String

Private mstrFileName As String
Private mintHFile As Integer
Private mstrFileHeader As String
Private mstrProjectName As String

Public Sub CloseFile()

   ' Close the file
   If mintHFile > 0 Then
      Call CloseFileSM(mstrFileName)
      mintHFile = 0
   End If

End Sub

Public Property Let ProjectName(ByVal vdata As String)
   ' An optional field - will be appended to the file
   ' header string if specified

   Const strProjectHdr As String = "Project Name:"

   mstrProjectName = vdata
   mstrFileHeader = mstrFileHeader & _
         Space$(1) & strProjectHdr & Space$(1) & _
         gstrSQ & vdata & gstrSQ

End Property
Public Property Get ProjectName() As String

   ProjectName = mstrProjectName

End Property
Public Property Get FileName() As String

   FileName = mstrFileName

End Property
Public Property Let FileName(ByVal vdata As String)

   mstrFileName = vdata

End Property
Public Sub WriteLine(strMsg As String)

   ' Writes the passed in string to the file
   Call WriteToFile(strMsg, False)

End Sub

Public Sub WriteField(strMsg As String)

   ' Writes the passed in string to the file
   Call WriteToFile(strMsg, True)

End Sub

Private Sub WriteToFile(strMsg As String, _
      blnContinue As Boolean)
   ' Writes the passed in string to the file - the
   ' Continue flag indicates whether the next line will
   ' be continued on the same line or printed on a new one

   On Error GoTo WriteToFileErr

   ' Open the file if it hasn't been already
   If mintHFile = 0 Then

      ' If the filename has not been initialized, do not
      ' attempt to open it
      If mstrFileName <> gstrEmptyString Then

         mintHFile = OpenFileSM(mstrFileName)
         If mintHFile = 0 Then
            ' The Open File command failed for some reason
            ' No point in trying to write the file header
         Else
            ' Print a file header, if a header string has been
            ' initialized
            If mstrFileHeader <> gstrEmptyString Then
               Print #mintHFile,
               Print #mintHFile, mstrFileHeader
               Print #mintHFile,
            End If
         End If
      End If
   End If

   If mintHFile <> 0 Then
      If strMsg = gstrEmptyString Then
         Print #mintHFile,
      Else
         If blnContinue Then
            ' Write the message to the file - continue
            ' all subsequent characters on the same line
            Print #mintHFile, strMsg;
         Else
            ' Write the message to the file
            Print #mintHFile, strMsg
         End If
      End If
   Else
      ' Display the string to the user instead of
      ' trying to write it to the file
      ' This could be the project error log that we were
      ' trying to open! Play it safe and display errors - do
      ' not try to log them.
      MsgBox strMsg, vbOKOnly
   End If

   Exit Sub

WriteToFileErr:
   ' Log the error code raised by Visual Basic
   Call DisplayErrors(Errors)

   ' Display the string to the user instead of
   ' trying to write it to the file
   MsgBox strMsg, vbOKOnly

End Sub
Public Property Let FileHeader(ByVal vdata As String)

   mstrFileHeader = vdata

End Property
Public Property Get FileHeader() As String

   FileHeader = mstrFileHeader

End Property

Private Sub Class_Terminate()

   ' Close the file opened by this instance
   Call CloseFile

End Sub
```

cGlobalStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
```

```
Attribute VB_Name = "cGlobalStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cGlobalStep.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of a global step.
'              Implements the cStep class - carries out initializations
'              and validations that are specific to global steps.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Implements cStep

' Object variable to keep the reference in
Private mcStep As cStep

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cGlobalStep."

Private Sub cStep_AddAllIterators()

   Call mcStep.AddAllIterators

End Sub

Private Sub cStep_AddIterator(cItRecord As cIterator)

   Call mcStep.AddIterator(cItRecord)

End Sub

Private Property Let cStep_ArchivedFlag(ByVal RHS As Boolean)

   mcStep.ArchivedFlag = RHS

End Property

Private Property Get cStep_ArchivedFlag() As Boolean

   cStep_ArchivedFlag = mcStep.ArchivedFlag

End Property

Private Sub Class_Initialize()

   ' Create the object
   Set mcStep = New cStep

   ' Initialize the object with valid values for a global step
   ' The global flag should be the first field to be initialized
   ' since subsequent validations might try to check if the
   ' step being created is global
   mcStep.GlobalFlag = True
   mcStep.StepType = gintGlobalStep

   ' A global step cannot have any sub-steps associated with it
   ' Hence, it will always be at Step Level 0
   mcStep.ParentStepId = 0
   mcStep.ParentVersionNo = gstrMinVersion
   mcStep.StepLevel = 0

   ' The enabled flag must be False for all global steps
   ' Global steps can be of two types
   ' a. Those that are run globally within a workspace either
   '    before every step, after every step or during the entire
   '    run, depending on the global run method
   ' b. Those that are not run globally, but qualify to be either
   '    pre or post-execution steps for other steps in the workspace.
   '    Whether or not such a step will be executed depends on
   '    whether the step for which it is defined as a pre/post
   '    step will be executed
   mcStep.EnabledFlag = False

   mcStep.ContinuationCriteria = gintNoOption
   mcStep.DegreeParallelism = gstrGlobalParallelism

End Sub
Private Sub Class_Terminate()

   ' Remove the step object
   Set mcStep = Nothing

End Sub

Private Sub cStep_Add()

   ' Call a private procedure to see if the step text has been
   ' entered - since a global step actually executes a step, entry
   ' of the text is mandatory
   Call StepTextOrFileEntered

   ' Call the Add method of the step class to carry out the insert
   mcStep.Add

End Sub

Private Function cStep_Clone(Optional cCloneStep As cStep) As cStep

   Dim cNewGlobal As cGlobalStep

   Set cNewGlobal = New cGlobalStep
   Set cStep_Clone = mcStep.Clone(cNewGlobal)

End Function


Private Property Get cStep_ContinuationCriteria() As ContinuationCriteria

   cStep_ContinuationCriteria = mcStep.ContinuationCriteria

End Property

Private Property Let cStep_ContinuationCriteria(ByVal RHS As
ContinuationCriteria)

   ' The continuation criteria field will always be empty for a
   ' global step
   mcStep.ContinuationCriteria = 0

End Property

Private Property Let cStep_DegreeParallelism(ByVal RHS As String)

   ' Will always be zero for a global step
   mcStep.DegreeParallelism = gstrGlobalParallelism

End Property

Private Property Get cStep_DegreeParallelism() As String

   cStep_DegreeParallelism = mcStep.DegreeParallelism

End Property

Private Sub cStep_DeleteIterator(cItRecord As cIterator)

   Call mcStep.DeleteIterator(cItRecord)

End Sub
```

```
Private Sub cStep_Delete()

   mcStep.Delete

End Sub

Private Property Get cStep_EnabledFlag() As Boolean

   cStep_EnabledFlag = mcStep.EnabledFlag

End Property

Private Property Let cStep_EnabledFlag(ByVal RHS As Boolean)

   ' The enabled flag must be False for all global steps
   ' Global steps can be of two types
   ' a. Those that are run globally within a workspace either
   '    before every step, after every step or during the entire
   '    run, depending on the global run method
   ' b. Those that are not run globally, but qualify to be either
   '    pre or post-execution steps for other steps in the workspace.
   '    Whether or not such a step will be executed depends on
   '    whether the step for which it is defined as a pre/post
   '    step will be executed
   mcStep.EnabledFlag = False

End Property

Private Property Let cStep_ErrorFile(ByVal RHS As String)

   mcStep.ErrorFile = RHS

End Property

Private Property Get cStep_ErrorFile() As String

   cStep_ErrorFile = mcStep.ErrorFile

End Property

Private Property Let cStep_ExecutionMechanism(ByVal RHS As
ExecutionMethod)

   ' Whether or not the Execution Mechanism is valid will be
   ' checked by the Step class
   mcStep.ExecutionMechanism = RHS

End Property

Private Property Get cStep_ExecutionMechanism() As ExecutionMethod

   cStep_ExecutionMechanism = mcStep.ExecutionMechanism

End Property

Private Property Let cStep_FailureDetails(ByVal RHS As String)

   ' Whether or not the Failure Details are valid for the
   ' selected failure criteria will be checked by the Step class
   mcStep.FailureDetails = RHS

End Property

Private Property Get cStep_FailureDetails() As String

   cStep_FailureDetails = mcStep.FailureDetails

End Property

Private Property Get cStep_GlobalFlag() As Boolean

   cStep_GlobalFlag = mcStep.GlobalFlag

End Property

Private Property Let cStep_GlobalFlag(ByVal RHS As Boolean)

   ' Set the global flag to true
   mcStep.GlobalFlag = True

End Property

Private Function cStep_IncVersionX() As String

   cStep_IncVersionX = mcStep.IncVersionX

End Function

Private Function cStep_IncVersionY() As String

   cStep_IncVersionY = mcStep.IncVersionY

End Function


'Private Property Let cStep_GlobalRunMethod(ByVal RHS As Integer)
'
'    ' Whether or not the Global Run Method is valid for the step
'    ' will be checked by the Step class
'    mcStep.GlobalRunMethod = RHS
'
'End Property
'
'Private Property Get cStep_GlobalRunMethod() As Integer
'
'    cStep_GlobalRunMethod = mcStep.GlobalRunMethod
'
'End Property
'
Private Property Get cStep_IndOperation() As Operation

   cStep_IndOperation = mcStep.IndOperation

End Property

Private Property Let cStep_IndOperation(ByVal RHS As Operation)

   mcStep.IndOperation = RHS

End Property

Private Sub cStep_InsertIterator(cItRecord As cIterator)

   Call mcStep.InsertIterator(cItRecord)

End Sub

Private Function cStep_IsNewVersion() As Boolean
   cStep_IsNewVersion = mcStep.IsNewVersion
End Function

Private Function cStep_IteratorCount() As Long

   cStep_IteratorCount = mcStep.IteratorCount

End Function

Private Property Let cStep_IteratorName(ByVal RHS As String)

   mcStep.IteratorName = RHS

End Property

Private Property Get cStep_IteratorName() As String

   cStep_IteratorName = mcStep.IteratorName
```

```
End Property

Private Function cStep_Iterators() As Variant

   cStep_Iterators = mcStep.Iterators

End Function

Private Sub cStep_LoadIterator(cItRecord As cIterator)

   Call mcStep.LoadIterator(cItRecord)

End Sub

'Private Property Let cStep_LogFile(ByVal RHS As String)
'
'    mcStep.LogFile = RHS
'
'End Property
'
'Private Property Get cStep_LogFile() As String
'
'    cStep_LogFile = mcStep.LogFile
'
'End Property

Private Sub cStep_ModifyIterator(cItRecord As cIterator)

   Call mcStep.ModifyIterator(cItRecord)

End Sub

Private Sub cStep_Modify()

   ' Call a private procedure to see if the step text has been
   ' entered - since a global step actually executes a step,
   ' entry of the text is mandatory
   Call StepTextOrFileEntered

   ' Call the Modify method of the step class to carry out the update
   mcStep.Modify

End Sub

Private Property Get cStep_NextStepId() As Long

   cStep_NextStepId = mcStep.NextStepId

End Property

Private Property Set cStep_NodeDB(RHS As DAO.Database)

   Set mcStep.NodeDB = RHS

End Property

Private Property Get cStep_NodeDB() As DAO.Database

   Set cStep_NodeDB = mcStep.NodeDB

End Property

Private Function cStep_OldVersionNo() As String
   cStep_OldVersionNo = mcStep.OldVersionNo
End Function

Private Property Let cStep_OutputFile(ByVal RHS As String)

   mcStep.OutputFile = RHS

End Property

Private Property Get cStep_OutputFile() As String

   cStep_OutputFile = mcStep.OutputFile

End Property

Private Property Let cStep_ParentStepId(ByVal RHS As Long)

   ' A global step cannot have any sub-steps associated with it
   ' Hence, the parent step id and parent version number will be zero
   mcStep.ParentStepId = 0

End Property

Private Property Get cStep_ParentStepId() As Long

   cStep_ParentStepId = mcStep.ParentStepId

End Property

Private Property Let cStep_ParentVersionNo(ByVal RHS As String)

   ' A global step cannot have any sub-steps associated with it
   ' Hence, the parent step id and parent version number will be zero
   mcStep.ParentVersionNo = gstrMinVersion

End Property

Private Property Get cStep_ParentVersionNo() As String

   cStep_ParentVersionNo = mcStep.ParentVersionNo

End Property

Private Property Let cStep_Position(ByVal RHS As Long)

   mcStep.Position = RHS

End Property

Private Property Get cStep_Position() As Long

   cStep_Position = mcStep.Position

End Property

Private Sub cStep_RemoveIterator(cItRecord As cIterator)

   Call mcStep.RemoveIterator(cItRecord)

End Sub

Private Sub cStep_SaveIterators()

   Call mcStep.SaveIterators

End Sub

Private Property Let cStep_SequenceNo(ByVal RHS As Integer)

   mcStep.SequenceNo = RHS

End Property

Private Property Get cStep_SequenceNo() As Integer

   cStep_SequenceNo = mcStep.SequenceNo

End Property

Private Property Let cStep_StepId(ByVal RHS As Long)

   mcStep.StepId = RHS

End Property
Private Property Get cStep_StepId() As Long
```

```
    cStep_StepId = mcStep.StepId

End Property


Private Property Let cStep_StepLabel(ByVal RHS As String)

    mcStep.StepLabel = RHS

End Property

Private Property Get cStep_StepLabel() As String

    cStep_StepLabel = mcStep.StepLabel

End Property

Private Property Let cStep_StartDir(ByVal RHS As String)

    mcStep.StartDir = RHS

End Property

Private Property Get cStep_StartDir() As String

    cStep_StartDir = mcStep.StartDir

End Property

Private Property Let cStep_StepLevel(ByVal RHS As Integer)

    ' A global step cannot have any sub-steps associated with it
    ' Hence, it will always be at step level 0
    mcStep.StepLevel = 0

End Property

Private Property Get cStep_StepLevel() As Integer

    cStep_StepLevel = mcStep.StepLevel

End Property

Private Property Let cStep_StepText(ByVal RHS As String)

    mcStep.StepText = RHS

End Property

Private Property Get cStep_StepText() As String

    cStep_StepText = mcStep.StepText

End Property

Private Property Let cStep_StepTextFile(ByVal RHS As String)

    mcStep.StepTextFile = RHS

End Property

Private Property Get cStep_StepTextFile() As String

    cStep_StepTextFile = mcStep.StepTextFile

End Property

Private Property Let cStep_StepType(RHS As gintStepType)

    mcStep.StepType = gintGlobalStep

End Property

Private Property Get cStep_StepType() As gintStepType

    cStep_StepType = mcStep.StepType

End Property

Private Sub cStep_UnloadIterators()

    Call mcStep.UnloadIterators

End Sub

Private Sub cStep_UpdateIterator(cItRecord As cIterator)

    Call mcStep.UpdateIterator(cItRecord)

End Sub

Private Sub cStep_UpdateIteratorVersion()

    Call mcStep.UpdateIteratorVersion

End Sub

Private Sub cStep_Validate()
    ' The validate routines for each of the steps will
    ' carry out the specific validations for the type and
    ' call the generic validation routine

    On Error GoTo cStep_ValidateErr
    mstrSource = mstrModuleName & "cStep_Validate"

    ' Validations specific to global steps

    ' Check if the step text or a file name has been
    ' specified
    Call StepTextOrFileEntered

    ' The step level must be zero for all globals
    If mcStep.StepLevel <> 0 Then
        ShowError errStepLevelZeroForGlobal
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.EnabledFlag Then
        ShowError errEnabledFlagFalseForGlobal
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.DegreeParallelism > 0 Then
        ShowError errDegParallelismNullForGlobal
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    If mcStep.ContinuationCriteria > 0 Then
        ShowError errContCriteriaNullForGlobal
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                gstrSource, _
                LoadResString(errValidateFailed)
    End If

    mcStep.Validate

    Exit Sub
```

```
cStep_ValidateErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "cStep_Validate"
   On Error GoTo 0
   Err.Raise vbObjectError + errValidateFailed, _
         mstrSource, _
         LoadResString(errValidateFailed)
End Sub
Private Sub StepTextOrFileEntered()
   ' Checks if either the step text or the name of the file containing
   ' the text has been entered
   ' If both of them are null or both of them are not null,
   ' the global step is invalid and an error is raised

   If StringEmpty(mcStep.StepText) And StringEmpty(mcStep.StepTextFile)
Then
      ShowError errStepTextAndFileNull
      On Error GoTo 0
      Err.Raise vbObjectError + errStepTextAndFileNull, _
         mstrSource, LoadResString(errStepTextAndFileNull)
   ElseIf Not StringEmpty(mcStep.StepText) And Not
StringEmpty(mcStep.StepTextFile) Then
      ShowError errStepTextOrFile
      On Error GoTo 0
      Err.Raise vbObjectError + errStepTextOrFile, _
         mstrSource, LoadResString(errStepTextOrFile)
   End If

End Sub

Private Property Let cStep_VersionNo(ByVal RHS As String)

   mcStep.VersionNo = RHS

End Property

Private Property Get cStep_VersionNo() As String

   cStep_VersionNo = mcStep.VersionNo

End Property

Private Property Let cStep_WorkspaceId(ByVal RHS As Long)

   mcStep.WorkspaceId = RHS

End Property

Private Property Get cStep_WorkspaceId() As Long

   cStep_WorkspaceId = mcStep.WorkspaceId

End Property
```

cInstance.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cInstance"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cInstance.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'  PURPOSE:    Encapsulates the properties and methods of an instance.
'              An instance is created when a step is executed for a
'              particular iterator value (if applicable) at 'run' time.
'              Contains functions to determine if an instance is running,
'              complete, and so on.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cInstance."
Private mstrSource As String

Private mcStep As cStep
Public Key As String ' Node key for the step being executed
Public InstanceId As Long
Public ParentInstanceId As Long ' The parent instance
Private mblnNoMoreToStart As Boolean
Private mblnComplete As Boolean
Public StartTime As Currency
Public EndTime As Currency
Public ElapsedTime As Currency
Private mintStatus As InstanceStatus
Public DegreeParallelism As Integer
Private mcIterators As cRunColIt

' A collection of all the sub-steps for this step
Private mcSubSteps As cSubSteps
Public Sub UpdateStartTime(lStepId As Long, Optional ByVal StartTm As
Currency = gdtmEmpty, _
      Optional ByVal EndTm As Currency = gdtmEmpty, _
      Optional ByVal Elapsed As Currency = 0)
   ' We do not maintain start and end timestamps for the constraint
   ' of a step. Hence we check if the process that just started/
   ' terminated is the worker step that is being executed. If so,
   ' we update the start/end time and status on the instance record.

   BugAssert (StartTm <> gdtmEmpty) Or (EndTm <> gdtmEmpty), "Mandatory
parameter missing."

   ' Make sure that we are executing the actual step and not
   ' a pre or post-execution constraint
   If mcStep.StepId = lStepId Then
      If StartTm <> 0 Then
         StartTime = StartTm
         mintStatus = gintRunning
      Else
         EndTime = EndTm
         ElapsedTime = Elapsed
         mintStatus = gintComplete
      End If
   End If

End Sub
Public Function ValidForIteration(cParentInstance As cInstance, _
      ByVal intConsType As ConstraintType) As Boolean
   ' Returns true if the instance passed in is the first or
   ' last iteration for the step, depending on the constraint type

   Dim cSubStepRec As cSubStep
   Dim vntIterators As Variant

   On Error GoTo ValidForIterationErr

   If cParentInstance Is Nothing Then
      ' This will only be true for the dummy instance, which
      ' cannot have any iterators defined for it
      ValidForIteration = True
      Exit Function
   End If

   vntIterators = mcStep.Iterators
```

```
    If Not StringEmpty(mcStep.IteratorName) And Not IsEmpty(vntIterators) Then

        Set cSubStepRec = cParentInstance.QuerySubStep(mcStep.StepId)

        If intConsType = gintPreStep Then
            ' Pre-execution constraints will only be executed
            ' before the first iteration
            If cSubStepRec.LastIterator.IteratorType = gintValue Then
                ValidForIteration = (cSubStepRec.LastIterator.Sequence = _
                        gintMinIteratorSequence)
            Else
                ValidForIteration = (cSubStepRec.LastIterator.Value = _
                        cSubStepRec.LastIterator.RangeFrom)
            End If
        Else
            ' Post-execution constraints will only be executed
            ' after the last iteration - check if there are any
            ' pending iterations
            ValidForIteration = cSubStepRec.NextIteration(mcStep) Is Nothing
        End If
    Else
        ValidForIteration = True
    End If

    Exit Function

ValidForIterationErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "ValidForIteration"
    Err.Raise vbObjectError + errExecInstanceFailed, _
            mstrSource, LoadResString(errExecInstanceFailed)

End Function

Public Sub CreateSubStep(cSubStepDtls As cStep, RunParams As
cArrParameters)

    Dim cNewSubStep As cSubStep

    On Error GoTo CreateSubStepErr

    Set cNewSubStep = New cSubStep

    cNewSubStep.StepId = cSubStepDtls.StepId
    cNewSubStep.TasksComplete = 0
    cNewSubStep.TasksRunning = 0

    ' Initialize the iterator for the instance
    Set cNewSubStep.LastIterator = New cRunItDetails
    Call cNewSubStep.InitializeIt(cSubStepDtls, RunParams)

    ' Add add the substep to the collection
    mcSubSteps.Add cNewSubStep

    Set cNewSubStep = Nothing

    Exit Sub

CreateSubStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "CreateSubStep"
    Err.Raise vbObjectError + errProgramError, mstrSource, _
        LoadResString(errProgramError)

End Sub

Public Function QuerySubStep(ByVal SubStepId As Long) As cSubStep
    ' Retrieves the sub-step record for the passed in sub-step id

    Dim lngIndex As Long

    On Error GoTo QuerySubStepErr

    ' Find the sub-step node with the matching step id
    For lngIndex = 0 To mcSubSteps.Count - 1
        If mcSubSteps(lngIndex).StepId = SubStepId Then
            Set QuerySubStep = mcSubSteps(lngIndex)
            Exit For
        End If
    Next lngIndex

    Exit Function

QuerySubStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "QuerySubStep"
    Err.Raise vbObjectError + errNavInstancesFailed, _
            mstrSource, LoadResString(errNavInstancesFailed)

End Function
Public Property Let AllStarted(ByVal vdata As Boolean)

    'bugmessage "Set All Started to " & vData & " for : " & _
        mstrKey

    mblnNoMoreToStart = vdata

End Property
Public Property Get AllStarted() As Boolean

    AllStarted = mblnNoMoreToStart

End Property
Public Property Let AllComplete(ByVal vdata As Boolean)

    'bugmessage "Set All Complete to " & vData & " for : " & _
        mstrKey

    mblnComplete = vdata

End Property

Public Property Get AllComplete() As Boolean

    AllComplete = mblnComplete

End Property

Public Sub ChildExecuted(mlngStepId As Long)
    ' This procedure is called when a sub-step executes.

    Dim lngIndex As Long

    On Error GoTo ChildExecutedErr

    BugAssert mcStep.StepType = gintManagerStep

    For lngIndex = 0 To mcSubSteps.Count - 1
        If mcSubSteps(lngIndex).StepId = mlngStepId Then
            mcSubSteps(lngIndex).TasksRunning = _
                    mcSubSteps(lngIndex).TasksRunning + 1
'           BugMessage "Tasks Running for Step Id : " & _
'                   CStr(mcSubSteps(lngIndex).StepId) & _
'                   " Instance Id: " & InstanceId & _
'                   " = " & mcSubSteps(lngIndex).TasksRunning
            Exit For
        End If
    Next lngIndex

    If lngIndex > mcSubSteps.Count - 1 Then
        ' The child step wasn't found - raise an error
        On Error GoTo 0
```

```
      Err.Raise vbObjectError + errInvalidChild, mstrModuleName, _
            LoadResString(errInvalidChild)
   End If

   Exit Sub

ChildExecutedErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errInstanceOpFailed, mstrModuleName &
"ChildExecuted", _
         LoadResString(errInstanceOpFailed)

End Sub
Public Sub ChildTerminated(mlngStepId As Long)
   ' This procedure is called when any sub-step process
   ' terminates. Note: The TasksComplete field will be
   ' updated only when all the instances for a sub-step
   ' complete execution.
   Dim lngIndex As Long

   On Error GoTo ChildTerminatedErr

   BugAssert mcStep.StepType = gintManagerStep

   For lngIndex = 0 To mcSubSteps.Count - 1

      If mcSubSteps(lngIndex).StepId = mlngStepId Then
         mcSubSteps(lngIndex).TasksRunning = _
               mcSubSteps(lngIndex).TasksRunning - 1
'          BugMessage "Tasks Running for Step Id : " & _
'                CStr(mcSubSteps(lngIndex).StepId) & _
'                " Instance Id: " & InstanceId & _
'                " = " & mcSubSteps(lngIndex).TasksRunning

         BugAssert mcSubSteps(lngIndex).TasksRunning >= 0, _
               "Tasks running for " & CStr(mlngStepId) & _
               " Instance Id " & InstanceId & " is less than 0."
         Exit For
      End If
   Next lngIndex

   If lngIndex > mcSubSteps.Count - 1 Then
      ' The child step wasn't found - raise an error
      On Error GoTo 0
      Err.Raise errInvalidChild, mstrModuleName & "ChildTerminated", _
            LoadResString(errInvalidChild)
   End If

   Exit Sub

ChildTerminatedErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   mstrSource = mstrModuleName & "ChildTerminated"
   Err.Raise vbObjectError + errInstanceOpFailed, mstrSource, _
         LoadResString(errInstanceOpFailed)

End Sub
Public Sub ChildCompleted(mlngStepId As Long)
   ' This procedure is called when any a sub-step completes
   ' execution. Note: The TasksComplete field will be
   ' incremented.
   Dim lngIndex As Long

   On Error GoTo ChildCompletedErr

   BugAssert mcStep.StepType = gintManagerStep

   For lngIndex = 0 To mcSubSteps.Count - 1
      BugAssert mcSubSteps(lngIndex).TasksComplete >= 0, _
            "Tasks complete for " & CStr(mcSubSteps(lngIndex).StepId) & _
            " Instance Id " & InstanceId & " is less than 0."

      If mcSubSteps(lngIndex).StepId = mlngStepId Then
         mcSubSteps(lngIndex).TasksComplete = _
               mcSubSteps(lngIndex).TasksComplete + 1
'          BugMessage "Tasks Complete for Step Id : " & _
'                CStr(mcSubSteps(lngIndex).StepId) & _
'                " Instance Id: " & InstanceId & _
'                " = " & mcSubSteps(lngIndex).TasksComplete
         Exit For
      End If
   Next lngIndex

   If lngIndex > mcSubSteps.Count - 1 Then
      ' The child step wasn't found - raise an error
      On Error GoTo 0
      Err.Raise errInvalidChild, mstrModuleName, _
            LoadResString(errInvalidChild)
   End If

   Exit Sub

ChildCompletedErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errInstanceOpFailed, mstrModuleName &
"ChildCompleted", _
         LoadResString(errInstanceOpFailed)

End Sub
Public Sub ChildDeleted(mlngStepId As Long)
   ' This procedure is called when a sub-step needs to be re-executed
   ' Note: The TasksComplete field is decremented. We needn't worry about
   ' the TasksRunning field since no steps are currently running.
   Dim lngIndex As Long

   On Error GoTo ChildDeletedErr

   BugAssert mcStep.StepType = gintManagerStep

   For lngIndex = 0 To mcSubSteps.Count - 1

      If mcSubSteps(lngIndex).StepId = mlngStepId Then
         mcSubSteps(lngIndex).TasksRunning = _
               mcSubSteps(lngIndex).TasksRunning - 1

         BugAssert mcSubSteps(lngIndex).TasksRunning >= 0, _
               "Tasks running for " & CStr(mcSubSteps(lngIndex).StepId) & _
               " Instance Id " & InstanceId & " is less than 0."
         Exit For
      End If
   Next lngIndex

   If lngIndex > mcSubSteps.Count - 1 Then
      ' The child step wasn't found - raise an error
      On Error GoTo 0
      Err.Raise errInvalidChild, mstrModuleName, _
            LoadResString(errInvalidChild)
   End If

   Exit Sub

ChildDeletedErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errInstanceOpFailed, mstrModuleName &
"ChildDeleted", _
         LoadResString(errInstanceOpFailed)

End Sub
Private Sub RaiseErrForWorker()
```

```
   If mcStep.StepType <> gintManagerStep Then
      On Error GoTo 0
      mstrSource = mstrModuleName & "RaiseErrForWorker"
      Err.Raise vbObjectError + errInvalidForWorker, _
            mstrSource, _
            LoadResString(errInvalidForWorker)
   End If

End Sub

Public Property Get Step() As cStep

   Set Step = mcStep

End Property
Public Property Get Iterators() As cRunColIt

   Set Iterators = mcIterators

End Property
Public Property Get SubSteps() As cSubSteps

   Call RaiseErrForWorker

   Set SubSteps = mcSubSteps

End Property
Public Property Set Step(cRunStep As cStep)

   Set mcStep = cRunStep

End Property

Public Property Set Iterators(cIts As cRunColIt)

   Set mcIterators = cIts

End Property
Public Property Get IsPending() As Boolean
   ' Returns true if the step has any substeps that need
   ' execution
   Dim lngIndex As Long
   Dim lngRunning As Long

   Call RaiseErrForWorker

   If Not mblnComplete And Not mblnNoMoreToStart Then
      ' Get a count of all the substeps that are already being
      ' executed
      lngRunning = 0
      For lngIndex = 0 To mcSubSteps.Count - 1
         lngRunning = lngRunning + mcSubSteps(lngIndex).TasksRunning
      Next lngIndex

      IsPending = (lngRunning < DegreeParallelism)
   Else
      ' This should be sufficient to prove that there r no
      ' more sub-steps to be executed.
      ' mblnComplete: Handles the case where all steps have
      ' been executed
      ' mblnNoMoreToStart: Handles the case where the step
      ' has a degree of parallelism greater than the total
      ' number of sub-steps available to execute
      IsPending = False
   End If

End Property
Public Property Get IsRunning() As Boolean
   ' Returns true if the any one of the substeps is still
   ' executing
   Dim lngIndex As Long

   Call RaiseErrForWorker

   IsRunning = False

   ' If a substep has no currently executing tasks and
   ' the tasks completed is greater than zero, then we can
   ' assume that it has completed execution (otherwise we
   ' would've run a new task the moment one completed!)
   For lngIndex = 0 To mcSubSteps.Count - 1
      If mcSubSteps(lngIndex).TasksRunning > 0 Then
         IsRunning = True
         Exit For
      End If
   Next lngIndex

End Property
Public Property Get TotalRunning() As Long
   ' Returns the total number of substeps that are executing
   Dim lngTotalProcesses As Long
   Dim lngIndex As Long

   Call RaiseErrForWorker

   lngTotalProcesses = 0
   For lngIndex = 0 To mcSubSteps.Count - 1
      BugAssert mcSubSteps(lngIndex).TasksRunning >= 0, _
            "Tasks running for " & CStr(mcSubSteps(lngIndex).StepId) & _
            " is less than 0."
      lngTotalProcesses = lngTotalProcesses +
mcSubSteps(lngIndex).TasksRunning
   Next lngIndex

   TotalRunning = lngTotalProcesses
End Property
Public Property Get RunningForStep(lngSubStepId As Long) As Long
   ' Returns the total number of instances of the substep
   ' that are executing
   Dim lngIndex As Long

   Call RaiseErrForWorker

   For lngIndex = 0 To mcSubSteps.Count - 1
      BugAssert mcSubSteps(lngIndex).TasksRunning >= 0, _
            "Tasks running for " & CStr(mcSubSteps(lngIndex).StepId) & _
            " is less than 0."

      If mcSubSteps(lngIndex).StepId = lngSubStepId Then
         RunningForStep = mcSubSteps(lngIndex).TasksRunning
         Exit For
      End If
   Next lngIndex

   If lngIndex > mcSubSteps.Count - 1 Then
      ' The child step wasn't found - raise an error
      On Error GoTo 0
      Err.Raise errInvalidChild, mstrSource, _
            LoadResString(errInvalidChild)
   End If

End Property

Public Property Let Status(ByVal vdata As InstanceStatus)

   mintStatus = vdata

End Property

Public Property Get Status() As InstanceStatus

   Status = mintStatus

End Property
Private Sub Class_Initialize()

   Set mcSubSteps = New cSubSteps
```

```
    mblnNoMoreToStart = False
    mblnComplete = False
    StartTime = gdtmEmpty
    EndTime = gdtmEmpty

End Sub

Private Sub Class_Terminate()

    mcSubSteps.Clear
    Set mcSubSteps = Nothing

End Sub
```

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

        If lngIndex > mcInstances.Count - 1 Then
            On Error GoTo 0
            Err.Raise vbObjectError + errQueryFailed, mstrSource, _
                  LoadResString(errQueryFailed)
        End If
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errQueryFailed, mstrSource, _
              LoadResString(errQueryFailed)
    End If

    Exit Function

QueryInstanceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "QueryInstance"
    Err.Raise vbObjectError + errQueryFailed, _
          mstrSource, LoadResString(errQueryFailed)

End Function

Public Function QueryPendingInstance(ByVal ParentInstanceId As Long, _
        ByVal lngSubStepId As Long) As cInstance
    ' Retrieves a pending instance for the passed in substep
    ' and the given parent instance id.

    Dim lngIndex As Long

    On Error GoTo QueryPendingInstanceErr

    ' Find the run node with the matching step id
    For lngIndex = 0 To Count() - 1
        If mcInstances(lngIndex).ParentInstanceId = ParentInstanceId And _
              mcInstances(lngIndex).Step.StepId = lngSubStepId Then
            ' Put in a separate if condition since the IsPending
            ' property is valid only for manager steps. If the
            ' calling procedure does not pass a manager step
            ' identifier, the procedure will error out.
            If mcInstances(lngIndex).IsPending Then
                Set QueryPendingInstance = mcInstances(lngIndex)
                Exit For
            End If
        End If
    Next lngIndex

    Exit Function

QueryPendingInstanceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "QueryPendingInstance"
    Err.Raise vbObjectError + errQueryFailed, _
          mstrSource, LoadResString(errQueryFailed)

End Function
Public Function InstanceAborted(cSubStepRec As cSubStep) As Boolean

    Dim lIndex As Long

    InstanceAborted = False

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

End Function
Public Function CompletedInstanceExists(lParentInstance As Long, _
        cSubStepDtls As cStep) As Boolean
    ' Checks if there is a completed instance of the passed in step

    Dim lngIndex As Long

    CompletedInstanceExists = False

    If cSubStepDtls.StepType = gintManagerStep Then
        ' Find the run node with the matching step id
        For lngIndex = 0 To Count() - 1
```

```
        If mcInstances(lngIndex).ParentInstanceId = lParentInstance And _
              mcInstances(lngIndex).Step.StepId = cSubStepDtls.StepId Then
            ' Put in a separate if condition since the IsPending
            ' property is valid only for manager steps.
            BugAssert (Not mcInstances(lngIndex).IsPending), "Pending instance
exists!"

            CompletedInstanceExists = True
            Exit Function
        End If
      Next lngIndex
   End If

End Function
Public Sub Add(ByVal objItem As cInstance)

   mcInstances.Add objItem

End Sub


Public Sub Clear()

   mcInstances.Clear

End Sub


Public Function Count() As Long

   Count = mcInstances.Count

End Function


Public Function Delete(ByVal lngDelete As Long) As cInstance

   Set Delete = mcInstances.Delete(lngDelete)

End Function


Public Property Set Item(Optional ByVal Position As Long, _
      RHS As cInstance)

   If Position = -1 Then
      Position = 0
   End If
   Set mcInstances(Position) = RHS

End Property

Public Property Get Item(Optional ByVal Position As Long = -1) _
      As cInstance
Attribute Item.VB_UserMemId = 0

   If Position = -1 Then
      Position = 0
   End If
   Set Item = mcInstances.Item(Position)

End Property

Private Sub Class_Initialize()

   Set mcInstances = New cVector

End Sub


Private Sub Class_Terminate()

   Set mcInstances = Nothing

End Sub
```

cIterator.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cIterator"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cIterator.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of an iterator.
'              Contains functions to insert, update and delete
'              iterator_values records from the database.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Implements cNode

' Module level variables to store the property values
Private mintType As Integer
Private mintSequenceNo As Integer
Private mstrValue As String
Private mdbsIteratorDB As Database
Private mintOperation As Integer
Private mlngPosition As Long

Private Const mstrModuleName As String = "cIterator."
Private mstrSource As String

Public Enum ValueType
   gintFrom = 1
   gintTo
   gintStep
   gintValue
End Enum
Public Property Get Value() As String

   Value = mstrValue

End Property
Public Property Let Value(ByVal vdata As String)

   mstrValue = vdata

End Property

Public Property Get IndOperation() As Operation

   IndOperation = mintOperation

End Property
Public Property Let IndOperation(ByVal vdata As Operation)

   On Error GoTo IndOperationErr
   mstrSource = mstrModuleName & "IndOperation"

   ' The valid operations are define in the cOperations
   ' class. Check if the operation is valid
   Select Case vdata
      Case QueryOp, InsertOp, UpdateOp, DeleteOp
         mintOperation = vdata

      Case Else
```

```
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidOperation, _
            mstrSource, LoadResString(errInvalidOperation)
    End Select

    Exit Property

IndOperationErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "IndOperation"
    On Error GoTo 0
    Err.Raise vbObjectError + errLetOperationFailed, _
      mstrSource, LoadResString(errLetOperationFailed)

End Property

Public Function Clone() As cIterator

    ' Creates a copy of a given Iterator

    Dim cItClone As cIterator

    On Error GoTo CloneErr

    Set cItClone = New cIterator

    ' Copy all the iterator properties to the newly
    ' created object
    cItClone.IteratorType = mintType
    cItClone.SequenceNo = mintSequenceNo
    cItClone.IndOperation = mintOperation
    cItClone.Value = mstrValue

    ' And set the return value to the newly created Iterator
    Set Clone = cItClone

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, _
        mstrSource, LoadResString(errCloneFailed)

End Function
Public Property Get SequenceNo() As Integer

    SequenceNo = mintSequenceNo

End Property

Public Property Let SequenceNo(ByVal vdata As Integer)
    mintSequenceNo = vdata
End Property

Public Sub Add(ByVal lngStepId As Long, _
      strVersion As String)
    ' Inserts a new iterator values record into the database

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddIteratorErr

    ' First check if the database object is valid
    Call CheckDB

    ' Create a temporary querydef object
    strInsert = "insert into iterator_values " & _
        "( step_id, version_no, type, " & _
        " iterator_value, sequence_no ) " & _
        " values ( [st_id], [ver_no], [it_typ], " & _
        " [it_val], [seq_no] )"
    Set qy = mdbsIteratorDB.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy, lngStepId, strVersion)

    qy.Execute dbFailOnError
    qy.Close

    Exit Sub

AddIteratorErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "AddIterator"
    On Error GoTo 0
    Err.Raise vbObjectError + errInsertIteratorFailed, _
        mstrSource, _
        LoadResString(errInsertIteratorFailed)
End Sub

Private Sub AssignParameters(qyExec As DAO.QueryDef, _
      ByVal lngStepId As Long, _
      strVersion As String)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the field names. When the parameter names are
    ' the same as the field names, parameters in the where
    ' clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr
    mstrSource = mstrModuleName & "AssignParameters"

    For Each prmParam In qyExec.Parameters
      Select Case prmParam.Name
        Case "[st_id]"
          prmParam.Value = lngStepId

        Case "[ver_no]"
          prmParam.Value = strVersion

        Case "[it_typ]"
          prmParam.Value = mintType

        Case "[it_val]"
          prmParam.Value = mstrValue

        Case "[seq_no]"
          prmParam.Value = mintSequenceNo

        Case Else
          ' Write the parameter name that is faulty
          WriteError errInvalidParameter, mstrSource, _
              prmParam.Name
          On Error GoTo 0
          Err.Raise errInvalidParameter, mstrSource, _
              LoadResString(errInvalidParameter)
      End Select
    Next prmParam

    Exit Sub

AssignParametersErr:

    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
      mstrSource, LoadResString(errAssignParametersFailed)

End Sub
Private Sub CheckDB()
    ' Check if the database object has been initialized
```

```
    If mdbsIteratorDB Is Nothing Then
        ShowError errInvalidDB
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidDB, _
            mstrModuleName, LoadResString(errInvalidDB)
    End If

End Sub

Public Sub Delete(ByVal lngStepId As Long, _
        strVersion As String)
    ' Deletes the step iterator record from the database

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteIteratorErr
    mstrSource = mstrModuleName & "DeleteIterator"

    ' There can be multiple iterators for a step.
    ' However the values that an iterator for a step can
    ' assume will be unique, meaning that a combination of
    ' the iterator_id and value will be unique.
    strDelete = "delete from iterator_values " & _
            " where step_id = [st_id]" & _
            " and version_no = [ver_no]" & _
            " and iterator_value = [it_val] "
    Set qy = mdbsIteratorDB.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy, lngStepId, strVersion)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteIteratorErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "DeleteIterator"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteIteratorFailed, _
            mstrSource, _
            LoadResString(errDeleteIteratorFailed)
End Sub
Public Sub Update(ByVal lngStepId As Long, strVersion As String)
    ' Updates the sequence no of the step iterator record
    ' in the database

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo UpdateErr

    ' First check if the database object is valid
    Call CheckDB

    If mintType = gintValue Then
        ' If the iterator is of type value, only the sequence of the values can get
updated
        strUpdate = "Update iterator_values " & _
                " set sequence_no = [seq_no] " & _
                " where step_id = [st_id]" & _
                " and version_no = [ver_no]" & _
                " and iterator_value = [it_val] "
    Else
        ' If the iterator is of type range, only the values can get updated
        strUpdate = "Update iterator_values " & _
                " set iterator_value = [it_val] " & _
                " where step_id = [st_id]" & _
                " and version_no = [ver_no]" & _
                " and type = [it_typ] "
    End If

    Set qy = mdbsIteratorDB.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the parameter values to the
    ' querydef object
    Call AssignParameters(qy, lngStepId, strVersion)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

UpdateErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Update"
    On Error GoTo 0
    Err.Raise vbObjectError + errUpdateConstraintFailed, _
            mstrSource, _
            LoadResString(errUpdateConstraintFailed)

End Sub
Public Property Set NodeDB(vdata As Database)

    Set mdbsIteratorDB = vdata

End Property

Public Property Get NodeDB() As Database

    Set NodeDB = mdbsIteratorDB

End Property
Public Property Get Position() As Long

    Position = mlngPosition

End Property
Public Property Let Position(ByVal vdata As Long)

    mlngPosition = vdata

End Property

Public Property Let IteratorType(ByVal vdata As ValueType)

    On Error GoTo TypeErr
    mstrSource = mstrModuleName & "Type"

    ' These constants have been defined in the enumeration,
    ' Type, which is exposed
    Select Case vdata
        Case gintFrom, gintTo, gintStep, gintValue
            mintType = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errTypeInvalid, _
                mstrSource, LoadResString(errTypeInvalid)
    End Select

    Exit Property

TypeErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Type"
    On Error GoTo 0
    Err.Raise vbObjectError + errTypeInvalid, _
        mstrSource, LoadResString(errTypeInvalid)

End Property

Public Property Get IteratorType() As ValueType

    IteratorType = mintType

End Property
```

```
Public Sub Validate()

   ' No validations necessary for the iterator class

End Sub

Private Sub Class_Initialize()

   ' Initialize the operation indicator variable to Query
   ' It will be modified later by the collection class when
   ' inserts, updates or deletes are performed
   mintOperation = QueryOp

End Sub


Private Property Let cNode_IndOperation(ByVal vdata As Operation)

   On Error GoTo IndOperationErr
   mstrSource = mstrModuleName & "IndOperation"

   ' The valid operations are define in the cOperations
   ' class. Check if the operation is valid
   Select Case vdata
      Case QueryOp, InsertOp, UpdateOp, DeleteOp
         mintOperation = vdata

      Case Else
         On Error GoTo 0
         Err.Raise vbObjectError + errInvalidOperation, _
               mstrSource, LoadResString(errInvalidOperation)
   End Select

   Exit Property

IndOperationErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "IndOperation"
   On Error GoTo 0
   Err.Raise vbObjectError + errLetOperationFailed, _
      mstrSource, LoadResString(errLetOperationFailed)

End Property

Private Property Get cNode_IndOperation() As Operation

   IndOperation = mintOperation

End Property

Private Property Set cNode_NodeDB(RHS As DAO.Database)

   Set mdbsIteratorDB = RHS

End Property

Private Property Get cNode_NodeDB() As DAO.Database

   Set cNode_NodeDB = mdbsIteratorDB

End Property


Private Property Let cNode_Position(ByVal vdata As Long)

   mlngPosition = vdata

End Property


Private Property Get cNode_Position() As Long

   cNode_Position = mlngPosition

End Property


Private Sub cNode_Validate()

   ' No validations necessary for the iterator class

End Sub


Private Property Let cNode_Value(ByVal vdata As String)

   mstrValue = vdata

End Property

Private Property Get cNode_Value() As String

   Value = mstrValue

End Property
```

cManager.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cManager"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cManager.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of a manager step.
'              Implements the cStep class - carries out initializations
'              and validations that are specific to manager steps.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Implements cStep

' Object variable to keep the step reference in
Private mcStep As cStep

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cManager."
Private Sub cStep_AddAllIterators()

   Call mcStep.AddAllIterators

End Sub

Private Property Let cStep_StartDir(ByVal RHS As String)

   mcStep.StartDir = RHS

End Property

Private Property Get cStep_StartDir() As String

   cStep_StartDir = mcStep.StartDir

End Property
Private Sub cStep_Delete()
```

```
    Call mcStep.Delete

End Sub

Private Property Set cStep_NodeDB(RHS As DAO.Database)

    Set mcStep.NodeDB = RHS

End Property

Private Function cStep_IncVersionY() As String

    cStep_IncVersionY = mcStep.IncVersionY

End Function
Private Function cStep_IsNewVersion() As Boolean
    cStep_IsNewVersion = mcStep.IsNewVersion
End Function
Private Function cStep_OldVersionNo() As String
    cStep_OldVersionNo = mcStep.OldVersionNo
End Function

Private Function cStep_IncVersionX() As String

    cStep_IncVersionX = mcStep.IncVersionX

End Function
Private Sub cStep_UpdateIteratorVersion()

    Call mcStep.UpdateIteratorVersion

End Sub

Private Function cStep_IteratorCount() As Long

    cStep_IteratorCount = mcStep.IteratorCount

End Function

Private Sub cStep_UnloadIterators()

    Call mcStep.UnloadIterators

End Sub

Private Sub cStep_DeleteIterator(cItRecord As cIterator)

    Call mcStep.DeleteIterator(cItRecord)

End Sub
Private Property Get cStep_IteratorName() As String

    cStep_IteratorName = mcStep.IteratorName

End Property
Private Property Let cStep_IteratorName(ByVal RHS As String)

    mcStep.IteratorName = RHS

End Property

Private Sub cStep_SaveIterators()

    Call mcStep.SaveIterators

End Sub
Private Sub cStep_LoadIterator(cItRecord As cIterator)

    Call mcStep.LoadIterator(cItRecord)

End Sub

Private Property Let cStep_Position(ByVal RHS As Long)

    mcStep.Position = RHS

End Property
Private Sub cStep_InsertIterator(cItRecord As cIterator)

    Call mcStep.InsertIterator(cItRecord)

End Sub
Private Function cStep_Iterators() As Variant

    cStep_Iterators = mcStep.Iterators

End Function
Private Sub cStep_ModifyIterator(cItRecord As cIterator)

    Call mcStep.ModifyIterator(cItRecord)

End Sub
Private Sub cStep_RemoveIterator(cItRecord As cIterator)

    Call mcStep.RemoveIterator(cItRecord)

End Sub
Private Sub cStep_UpdateIterator(cItRecord As cIterator)

    Call mcStep.UpdateIterator(cItRecord)

End Sub
Private Sub cStep_AddIterator(cItRecord As cIterator)

    Call mcStep.AddIterator(cItRecord)

End Sub

Private Property Get cStep_Position() As Long

    cStep_Position = mcStep.Position

End Property

Private Function cStep_Clone(Optional cCloneStep As cStep) As cStep

    Dim cNewManager As cManager

    Set cNewManager = New cManager
    Set cStep_Clone = mcStep.Clone(cNewManager)

End Function

Private Property Get cStep_IndOperation() As Operation

    cStep_IndOperation = mcStep.IndOperation

End Property

Private Property Let cStep_IndOperation(ByVal RHS As Operation)

    mcStep.IndOperation = RHS

End Property

Private Property Get cStep_NextStepId() As Long

    cStep_NextStepId = mcStep.NextStepId

End Property

Private Property Let cStep_OutputFile(ByVal RHS As String)

    mcStep.OutputFile = RHS

End Property
```

```
Private Property Get cStep_OutputFile() As String

   cStep_OutputFile = mcStep.OutputFile

End Property

Private Property Let cStep_ErrorFile(ByVal RHS As String)

   mcStep.ErrorFile = RHS

End Property

Private Property Get cStep_ErrorFile() As String

   cStep_ErrorFile = mcStep.ErrorFile

End Property
'Private Property Let cStep_LogFile(ByVal RHS As String)
'
'    mcStep.LogFile = RHS
'
'End Property
'
'Private Property Get cStep_LogFile() As String
'
'    cStep_LogFile = mcStep.LogFile
'
'End Property

Private Property Let cStep_ArchivedFlag(ByVal RHS As Boolean)

   mcStep.ArchivedFlag = RHS

End Property

Private Property Get cStep_ArchivedFlag() As Boolean

   cStep_ArchivedFlag = mcStep.ArchivedFlag

End Property

Private Property Get cStep_NodeDB() As DAO.Database

   Set cStep_NodeDB = mcStep.NodeDB

End Property

Private Sub Class_Initialize()

   ' Create the object
   Set mcStep = New cStep

   ' Initialize the object with valid values for a manager step
   ' The global flag should be the first field to be initialized
   ' since subsequent validations might try to check if the
   ' step being created is global
   mcStep.GlobalFlag = False
'    mcStep.GlobalRunMethod = gintNoOption
   mcStep.StepType = gintManagerStep

   ' Since the manager step does not take any action, the step
   ' text and file name will always be empty
   mcStep.StepText = gstrEmptyString
   mcStep.StepTextFile = gstrEmptyString

   ' Since the manager step does not take any action, execution
   ' properties for the step will be empty
   mcStep.ExecutionMechanism = gintNoOption
   mcStep.FailureDetails = gstrEmptyString
   mcStep.ContinuationCriteria = gintNoOption

End Sub
Private Sub Class_Terminate()

   ' Remove the step object
   Set mcStep = Nothing

End Sub
Private Sub cStep_Add()

   ' Call the Add method of the step class to carry out the insert
   mcStep.Add

End Sub
Private Property Get cStep_ContinuationCriteria() As ContinuationCriteria

   cStep_ContinuationCriteria = mcStep.ContinuationCriteria

End Property

Private Property Let cStep_ContinuationCriteria(ByVal RHS As
ContinuationCriteria)

   ' Since a manager step cannot take any action, the continuation
   ' criteria property does not apply to it
   mcStep.ContinuationCriteria = gintNoOption

End Property

Private Property Let cStep_DegreeParallelism(ByVal RHS As String)

   mcStep.DegreeParallelism = RHS

End Property

Private Property Get cStep_DegreeParallelism() As String

   cStep_DegreeParallelism = mcStep.DegreeParallelism

End Property

Private Sub cStep_DeleteStep()

   On Error GoTo cStep_DeleteStepErr
   mstrSource = mstrModuleName & "cStep_DeleteStep"

   mcStep.Delete
   Exit Sub

cStep_DeleteStepErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "cStep_DeleteStep"
   On Error GoTo 0
   Err.Raise vbObjectError + errDeleteStepFailed, _
         mstrSource, _
         LoadResString(errDeleteStepFailed)

End Sub

Private Property Get cStep_EnabledFlag() As Boolean

   cStep_EnabledFlag = mcStep.EnabledFlag

End Property

Private Property Let cStep_EnabledFlag(ByVal RHS As Boolean)

   mcStep.EnabledFlag = RHS

End Property

Private Property Let cStep_ExecutionMechanism(ByVal RHS As
ExecutionMethod)

   ' Since a manager step cannot take any action, the Execution
   ' Mechanism property does not apply to it
   mcStep.ExecutionMechanism = gintNoOption
```

```
End Property

Private Property Get cStep_ExecutionMechanism() As ExecutionMethod

   cStep_ExecutionMechanism = mcStep.ExecutionMechanism

End Property

Private Property Let cStep_FailureDetails(ByVal RHS As String)

   ' Since a manager step cannot take any action, the Failure
   ' Details property does not apply to it
   mcStep.FailureDetails = gstrEmptyString

End Property

Private Property Get cStep_FailureDetails() As String

   cStep_FailureDetails = mcStep.FailureDetails

End Property

Private Property Get cStep_GlobalFlag() As Boolean

   cStep_GlobalFlag = mcStep.GlobalFlag

End Property

Private Property Let cStep_GlobalFlag(ByVal RHS As Boolean)

   ' Set the global flag to false - this flag is initialized when
   ' an instance of the class is created. Just making sure that
   ' nobody changes the value inadvertently
   mcStep.GlobalFlag = False

End Property
Private Sub cStep_Modify()

   ' Call the Modify method of the step class to carry out the update
   mcStep.Modify

End Sub

Private Property Let cStep_ParentStepId(ByVal RHS As Long)

   mcStep.ParentStepId = RHS

End Property

Private Property Get cStep_ParentStepId() As Long

   cStep_ParentStepId = mcStep.ParentStepId

End Property

Private Property Let cStep_ParentVersionNo(ByVal RHS As String)

   mcStep.ParentVersionNo = RHS

End Property

Private Property Get cStep_ParentVersionNo() As String

   cStep_ParentVersionNo = mcStep.ParentVersionNo

End Property

Private Property Let cStep_SequenceNo(ByVal RHS As Integer)

   mcStep.SequenceNo = RHS

End Property

Private Property Get cStep_SequenceNo() As Integer

   cStep_SequenceNo = mcStep.SequenceNo

End Property

Private Property Let cStep_StepId(ByVal RHS As Long)

   mcStep.StepId = RHS

End Property

Private Property Get cStep_StepId() As Long

   cStep_StepId = mcStep.StepId

End Property


Private Property Let cStep_StepLabel(ByVal RHS As String)

   mcStep.StepLabel = RHS

End Property

Private Property Get cStep_StepLabel() As String

   cStep_StepLabel = mcStep.StepLabel

End Property


Private Property Let cStep_StepLevel(ByVal RHS As Integer)

   mcStep.StepLevel = RHS

End Property

Private Property Get cStep_StepLevel() As Integer

   cStep_StepLevel = mcStep.StepLevel

End Property

Private Property Let cStep_StepText(ByVal RHS As String)

   ' Since the manager step does not take any action, the step
   ' text and file name will always be empty
   mcStep.StepText = gstrEmptyString

End Property

Private Property Get cStep_StepText() As String

   cStep_StepText = mcStep.StepText

End Property

Private Property Let cStep_StepTextFile(ByVal RHS As String)

   ' Since the manager step does not take any action, the step
   ' text and file name will always be empty
   mcStep.StepTextFile = gstrEmptyString

End Property

Private Property Get cStep_StepTextFile() As String

   cStep_StepTextFile = mcStep.StepTextFile

End Property

Private Property Let cStep_StepType(RHS As gintStepType)

   mcStep.StepType = gintManagerStep
```

```
End Property

Private Property Get cStep_StepType() As gintStepType

   cStep_StepType = mcStep.StepType

End Property

Private Sub cStep_Validate()
   ' The validate routines for each of the steps will
   ' carry out the specific validations for the type and
   ' call the generic validation routine

   On Error GoTo cStep_ValidateErr
   mstrSource = mstrModuleName & "cStep_Validate"

   ' Validations specific to manager steps

   ' Check if the step text or a file name has been
   ' specified
   If Not StringEmpty(mcStep.StepText) Or Not
StringEmpty(mcStep.StepTextFile) Then
      ShowError errTextAndFileNullForManager
      On Error GoTo 0
      Err.Raise vbObjectError + errValidateFailed, _
            gstrSource, _
            LoadResString(errValidateFailed)
   End If

   If mcStep.ExecutionMechanism <> gintNoOption Then
      ShowError errExecutionMechanismInvalid
      On Error GoTo 0
      Err.Raise vbObjectError + errValidateFailed, _
            gstrSource, _
            LoadResString(errValidateFailed)
   End If

   If mcStep.FailureDetails <> gstrEmptyString Then
      ShowError errFailureDetailsNullForMgr
      On Error GoTo 0
      Err.Raise vbObjectError + errValidateFailed, _
            gstrSource, _
            LoadResString(errValidateFailed)
   End If

   If mcStep.ContinuationCriteria <> gintNoOption Then
      ShowError errContCriteriaInvalid
      On Error GoTo 0
      Err.Raise vbObjectError + errValidateFailed, _
            gstrSource, _
            LoadResString(errValidateFailed)
   End If

   mcStep.Validate

   Exit Sub

cStep_ValidateErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "cStep_Validate"
   On Error GoTo 0
   Err.Raise vbObjectError + errValidateFailed, _
         mstrSource, _
         LoadResString(errValidateFailed)
End Sub

Private Property Let cStep_VersionNo(ByVal RHS As String)

   mcStep.VersionNo = RHS

End Property

Private Property Get cStep_VersionNo() As String

   cStep_VersionNo = mcStep.VersionNo

End Property

Private Property Let cStep_WorkspaceId(ByVal RHS As Long)

   mcStep.WorkspaceId = RHS

End Property

Private Property Get cStep_WorkspaceId() As Long

   cStep_WorkspaceId = mcStep.WorkspaceId

End Property
```

cNode.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cNode"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cNode.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     Defines the properties that an object has to implement.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Property Get IndOperation() As Operation
End Property
Public Property Let IndOperation(ByVal vdata As Operation)
End Property
Public Sub Validate()
End Sub
Public Property Get Value() As String
End Property
Public Property Let Value(ByVal vdata As String)
End Property

Public Property Get NodeDB() As Database
End Property
Public Property Set NodeDB(vdata As Database)
End Property

Public Property Get Position() As Long
End Property
Public Property Let Position(ByVal vdata As Long)
End Property
```

cNodeCollections.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cNodeCollections"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
```

```
'  FILE:      cNodeCollections.cls
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Implements an array of objects.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Node counter
Private mlngNodeCount As Long
Private mdbsNodeDb As Database
Private mcarrNodes() As Object

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cNodeCollections."

Public Property Set Item(ByVal Position As Long, _
      ByVal objNode As Object)

   ' Returns the element at the passed in position in the array
   If Position >= 0 And Position < mlngNodeCount Then
      Set mcarrNodes(Position) = objNode
   Else
      On Error GoTo 0
      Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
         LoadResString(errItemDoesNotExist)
   End If

End Property
Public Property Get Item(ByVal Position As Long) As Object
Attribute Item.VB_UserMemId = 0

   ' Returns the element at the passed in position in the array
   If Position >= 0 And Position < mlngNodeCount Then
      Set Item = mcarrNodes(Position)
   Else
      On Error GoTo 0
      Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
         LoadResString(errItemDoesNotExist)
   End If

End Property

Public Sub Commit(ByVal cSaveObj As Object, _
      ByVal lngIndex As Long)
   ' This procedure checks if any changes have been made to the
   ' passed in object. If so, it calls the corresponding method
   ' to commit the changes.

   On Error GoTo CommitErr
   mstrSource = mstrModuleName & "Commit"

   Select Case cSaveObj.IndOperation
      Case QueryOp
         ' No changes were made to the queried parameter.
         ' Do nothing

      Case InsertOp
         cSaveObj.Add
         cSaveObj.IndOperation = QueryOp

      Case UpdateOp
         cSaveObj.Modify
         cSaveObj.IndOperation = QueryOp

      Case DeleteOp
         cSaveObj.Delete
         ' Now we can remove the record from the array
         Call Unload(lngIndex)

   End Select

   Exit Sub

CommitErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Commit"
   On Error GoTo 0
   Err.Raise vbObjectError + errCommitFailed, _
         mstrSource, _
         LoadResString(errCommitFailed)

End Sub
Public Sub Save(ByVal lngWorkspace As Long)
   ' Calls a procedure to commit all changes for the passed
   ' in workspace.

   Dim lngIndex As Long

   On Error GoTo SaveErr

   ' Find all parameters in the array with a matching workspace id
   ' It is important to step backwards through the array, since
   ' we delete parameter records as we go along!
   For lngIndex = mlngNodeCount - 1 To 0 Step -1
      If mcarrNodes(lngIndex).WorkspaceId = lngWorkspace Then

         ' Call a procedure to commit all changes to the
         ' parameter record, if any
         Call Commit(mcarrNodes(lngIndex), lngIndex)

      End If
   Next lngIndex

   Exit Sub

SaveErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Save"
   On Error GoTo 0
   Err.Raise vbObjectError + errSaveFailed, _
         mstrSource, _
         LoadResString(errSaveFailed)

End Sub
Public Property Get Count() As Long

   Count = mlngNodeCount

End Property

Public Property Get NodeDB() As Database

   Set NodeDB = mdbsNodeDb

End Property
Public Property Set NodeDB(vdata As Database)

   Set mdbsNodeDb = vdata

End Property

Public Sub Load(cNodeToLoad As Object)
   ' Adds the passed in object to the array

   On Error GoTo LoadErr

   ' If this procedure is called by the add to array procedure,
   ' the database object has already been initialized
   If cNodeToLoad.NodeDB Is Nothing Then

      ' All the Nodes will be initialized with the database
      ' objects before being added to the array
```

```
        Set cNodeToLoad.NodeDB = mdbsNodeDb

    End If

    ReDim Preserve mcarrNodes(mlngNodeCount)

    ' Set the newly added element in the array to the passed in Node
    cNodeToLoad.Position = mlngNodeCount
    Set mcarrNodes(mlngNodeCount) = cNodeToLoad

    mlngNodeCount = mlngNodeCount + 1

    Exit Sub

LoadErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadFailed, mstrModuleName & "Load", _
            LoadResString(errLoadFailed)

End Sub
Public Sub Unload(lngDeletePosition As Long)
    ' Unloads the passed in object from the array

    On Error GoTo UnloadErr

    If lngDeletePosition < (mlngNodeCount - 1) Then

        ' Set the Node at the position being deleted to
        ' the last Node in the Node array
        Set mcarrNodes(lngDeletePosition) = mcarrNodes(mlngNodeCount - 1)
        mcarrNodes(lngDeletePosition).Position = lngDeletePosition
    End If

    ' Delete the last Node from the array
    mlngNodeCount = mlngNodeCount - 1
    If mlngNodeCount > 0 Then
        ReDim Preserve mcarrNodes(0 To mlngNodeCount - 1)
    Else
        ReDim mcarrNodes(0)
    End If

    Exit Sub

UnloadErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Unload"
    On Error GoTo 0
    Err.Raise vbObjectError + errUnloadFailed, _
            mstrSource, _
            LoadResString(errUnloadFailed)

End Sub
Public Sub Delete(lngDeletePosition As Long)
    ' Deletes the object at the specified position in the
    ' array

    Dim cDeleteObj As Object

    On Error GoTo DeleteErr
    mstrSource = mstrModuleName & "Delete"

    Set cDeleteObj = mcarrNodes(lngDeletePosition)

    If cDeleteObj.IndOperation = InsertOp Then
        ' If we are deleting a record that has just been inserted,
        ' blow it away
        Call Unload(lngDeletePosition)
    Else
        ' Set the operation for the deleted object to indicate a
        ' delete - we actually delete the element only at the time
        ' of a save operation
        cDeleteObj.IndOperation = DeleteOp
    End If

    Exit Sub

DeleteErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteFailed, _
            mstrSource, _
            LoadResString(errDeleteFailed)

End Sub
Public Sub Modify(cModifiedNode As Object)
    ' Sets the object at the passed in position to the
    ' modified object passed in

    On Error GoTo ModifyErr

    ' First check if the record is valid - all objects that
    ' use this collection class must have a Validate routine
    cModifiedNode.Validate

    ' If we are updating a record that hasn't yet been inserted,
    ' do not change the operation indicator - or we try to update
    ' a non-existant record
    If cModifiedNode.IndOperation <> InsertOp Then
        ' Set the operations to indicate an update
        cModifiedNode.IndOperation = UpdateOp
    End If

    ' Modify the object at the queried position - the Position
    ' will be maintained by this class
    Set mcarrNodes(cModifiedNode.Position) = cModifiedNode

    Exit Sub

ModifyErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Modify"
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyFailed, _
            mstrSource, _
            LoadResString(errModifyFailed)

End Sub
Public Sub Add(cNodeToAdd As Object)

    On Error GoTo AddErr

    Set cNodeToAdd.NodeDB = mdbsNodeDb

    ' First check if the record is valid
    cNodeToAdd.Validate

    ' Set the operation to indicate an insert
    cNodeToAdd.IndOperation = InsertOp

    ' Call a procedure to load the record in the array
    Call Load(cNodeToAdd)

    Exit Sub

AddErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Add"
    On Error GoTo 0
    Err.Raise vbObjectError + errAddFailed, _
            mstrSource, _
            LoadResString(errAddFailed)

End Sub

Private Sub Class_Terminate()
```

```
    ReDim mcarrNodes(0)
    mlngNodeCount = 0

End Sub
```

Common.bas

```
Attribute VB_Name = "Common"
'  FILE:      Common.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Module containing common functionality throughout
'             StepMaster
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private Const mstrModuleName As String = "Common."

' Used to separate the variable data from the constant error
' message being raised when a context-sensitive error is displayed
Private Const mintDelimiter As String = " : "
Private Const mstrFormatString = "mmddyy"

' Identifiers for the different labels that need to be loaded
' into the tree view for each workspace
Public Const mstrWorkspacePrefix = "W"
Public Const mstrParameterPrefix = "P"
Public Const mstrParamConnectionPrefix = "C"
Public Const mstrConnectionDtlPrefix = "N"
Public Const mstrParamExtensionPrefix = "E"
Public Const mstrParamBuiltInPrefix = "B"
Public Const gstrGlobalStepPrefix = "G"
Public Const gstrManagerStepPrefix = "M"
Public Const gstrWorkerStepPrefix = "S"
Public Const gstrDummyPrefix = "D"
Public Const mstrLabelPrefix = "L"
Public Const mstrInstancePrefix = "I"
Public Function LabelStep(lngWorkspaceIdentifier As Long) As String
    ' Returns the step label for the workspace identifier passed in
    ' Basically this is a wrapper around the MakeKeyValid function

    LabelStep = MakeKeyValid(gintStepLabel, gintStepLabel,
lngWorkspaceIdentifier)

End Function


Public Function JulianDateToString(dt64Bit As Currency) As String

    Dim lYear As Long
    Dim lMonth As Long
    Dim lDay As Long
    Dim lHour As Long
    Dim lMin As Long
    Dim lSec As Long
    Dim lMs As Long

    Call JulianToTime(dt64Bit, lYear, lMonth, lDay, lHour, lMin, lSec, lMs)
    JulianDateToString = Format$(lYear, gsYearFormat) & gsDateSeparator & _
            Format$(lMonth, gsDtFormat) & gsDateSeparator & _
            Format$(lDay, gsDtFormat) & gstrBlank & _
            Format$(lHour, gsTmFormat) & gsTimeSeparator & _
            Format$(lMin, gsTmFormat) & gsTimeSeparator & _
            Format$(lSec, gsTmFormat) & gsMsSeparator & _
            Format$(lMs, gsMSecondFormat)

End Function
Public Sub DeleteFile(strFile As String, Optional ByVal bCheckIfEmpty As
Boolean = False)

    ' Ensure that there is only a single file of the name before delete, since
    ' Kill supports wildcards and can potentially delete a number of files
    Dim strTemp As String

    If CheckFileExists(strFile) Then
        If bCheckIfEmpty Then
            If FileLen(strFile) = 0 Then
                Kill strFile
            End If
        Else
            Kill strFile
        End If
    End If

End Sub
Public Function CheckFileExists(strFile As String) As Boolean

    ' Returns true if the passed in file exists
    ' Raises an error if multiple files are found (filename contains a wildcard)
    CheckFileExists = False

    If Not StringEmpty(Dir(strFile)) Then
        If Not StringEmpty(Dir()) Then
            On Error GoTo 0
            Err.Raise vbObjectError + errDeleteSingleFile, _
                mstrModuleName & "DeleteFile", LoadResString(errDeleteSingleFile)
        End If

        CheckFileExists = True
    End If

End Function

Public Function GetVersionString() As String
GetVersionString = "Version " & gsVersion
End Function

Function IsLabel(strKey As String) As Boolean

    ' The tree view control on frmMain can contain two types of
    ' nodes -
    ' 1. Nodes that contain data for the workspace - this could
    ' be data for the different types of steps or parameters
    ' 2. Nodes that display static data - these kind of nodes
    ' are referred to as label nodes e.g. "Global Steps" is a
    ' label node
    ' This function returns True if the passed in key corresponds
    ' to a label node

    IsLabel = InStr(strKey, mstrLabelPrefix) > 0

End Function

Function MakeKeyValid(lngIdentifier As Long, _
    intTypeOfNode As Integer, _
    Optional ByVal WorkspaceId As Long = 0, _
    Optional ByVal InstanceId As Long = 0) As String

    ' We use a numbering scheme while loading the tree view with
    ' all node data, since it needs a unique key and we want to
    ' use the key to identify the data it contains.
    ' Moreover, add a character to the beginning of the identifier
    ' so that the tree view control accepts it as a valid string,
    ' viz. "456" doesn't work, so change it to "W456"
    ' The general scheme is to concatenate a Label with the Identifier
    ' e.g A Global Step Node will have the Label, G and the Step Id
    ' concatenated to form the unique key
    ' The list of all such node types is given below
    ' 1. "W" + Workspace_Id for Workspace nodes
    ' 2. "P" + Parameter_Id for Parameter nodes
    ' 3. "M" + Step_Id for Manager Step nodes
    ' 4. "S" + Step_Id for Worker Step nodes
    ' 5. "G" + Step_Id for Global Step nodes
```

```
    ' 6. Instance_id + "I" + Step_Id for Instance nodes
    ' 7. Workspace_id + "L" + the label identifier = node type for all Label nodes
    ' Since the manager, worker and global steps are stored in the
    ' same table and the step identifiers will always be unique, we
    ' can use the same character as the prefix, but this is a
    ' convenient way to know the type of step being processed.
    ' The workspace id is appended to the label identifier to make
    ' it unique, since multiple workspaces may be open during a session
    ' Strip the prefix characters off while saving the Ids to the db

    Dim strPrefixChar As String

    On Error GoTo MakeKeyValidErr
    gstrSource = mstrModuleName & "MakeKeyValid"

    Select Case intTypeOfNode
        Case gintWorkspace
            strPrefixChar = mstrWorkspacePrefix
        Case gintGlobalStep
            strPrefixChar = gstrGlobalStepPrefix
        Case gintManagerStep
            strPrefixChar = gstrManagerStepPrefix
        Case gintWorkerStep
            strPrefixChar = gstrWorkerStepPrefix
        Case gintRunManager, gintRunWorker
            If InstanceId = 0 Then
                On Error GoTo 0
                Err.Raise vbObjectError + errMandatoryParameterMissing, _
                    gstrSource, _
                    LoadResString(errMandatoryParameterMissing)
            End If
            ' Concatenate the instance identifier and the step
            ' identifier to form a unique key
            strPrefixChar = Trim$(Str$(InstanceId)) & mstrInstancePrefix
        Case gintParameter
            strPrefixChar = mstrParameterPrefix
        Case gintNodeParamConnection
            strPrefixChar = mstrParamConnectionPrefix
        Case gintConnectionDtl
            strPrefixChar = mstrConnectionDtlPrefix
        Case gintNodeParamExtension
            strPrefixChar = mstrParamExtensionPrefix
        Case gintNodeParamBuiltIn
            strPrefixChar = mstrParamBuiltInPrefix
        Case gintGlobalsLabel, gintParameterLabel, gintParamConnectionLabel, _
                gintConnDtlLabel, _
                gintParamExtensionLabel, gintParamBuiltInLabel,
gintGlobalStepLabel, _
                gintStepLabel
            If WorkspaceId = 0 Then
                ' The Workspace Id has to be specified for a label node
                ' Otherwise it will not be possible to generate unique label
                ' identifiers if multiple workspaces are open
                On Error GoTo 0
                Err.Raise vbObjectError + errWorkspaceIdMandatory, _
                    gstrSource, _
                    LoadResString(errWorkspaceIdMandatory)
            End If
            ' For all labels, the workspace identifier and the
            ' label prefix are concatenated to form the key
            strPrefixChar = Trim$(Str$(WorkspaceId)) & mstrLabelPrefix
        Case Else
                On Error GoTo 0
                Err.Raise vbObjectError + errInvalidNodeType, _
                    gstrSource, _
                    LoadResString(errInvalidNodeType)
    End Select

    MakeKeyValid = strPrefixChar & Trim$(Str$(lngIdentifier))

    Exit Function

MakeKeyValidErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errMakeKeyValidFailed, _
            gstrSource, _
            LoadResString(errMakeKeyValidFailed)

End Function

Function MakeIdentifierValid(strKey As String) As Long

    ' Returns the Identifier corresponding to the passed in key
    ' (Reverse of what was done in MakeKeyValid)

    On Error GoTo MakeIdentifierValidErr

    If IsLabel(strKey) Then
        ' If the key corresponds to a label node, the identifier
        ' appears to the right of the label prefix
        MakeIdentifierValid = Val(Mid(strKey, InStr(strKey, mstrLabelPrefix) + 1))
    ElseIf InStr(strKey, mstrInstancePrefix) = 0 Then
        ' For all other nodes, stripping the first character off
        ' returns a valid Id
        MakeIdentifierValid = Val(Mid(strKey, 2))
    Else
        ' Instance node - strip of all characters till the
        ' instance prefix
        MakeIdentifierValid = Val(Mid(strKey, InStr(strKey, mstrInstancePrefix) +
1))
    End If

    Exit Function

MakeIdentifierValidErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errMakeIdentifierValidFailed, _
            mstrModuleName & "MakeIdentifierValid", _
            LoadResString(errMakeIdentifierValidFailed)

End Function
Public Function IsInstanceNode(strNodeKey As String) As Boolean

    ' Returns true if the passed in node key corresponds to a step instance
    IsInstanceNode = InStr(strNodeKey, mstrInstancePrefix) > 0

End Function
Public Function IsBuiltInLabel(strNodeKey As String) As Boolean

    ' Returns true if the passed in node key corresponds to a step instance
    IsBuiltInLabel = (IsLabel(strNodeKey) And _
        (MakeIdentifierValid(strNodeKey) = gintParamBuiltInLabel))

End Function

Public Sub ShowBusy()
    ' Modifies the mousepointer to indicate that the
    ' application is busy

    On Error Resume Next

    Screen.MousePointer = vbHourglass

End Sub
Public Sub ShowFree()
    ' Modifies the mousepointer to indicate that the
    ' application has finished processing and is ready
    ' to accept user input

    On Error Resume Next

    Screen.MousePointer = vbDefault

End Sub

Public Function InstrR(strMain As String, _
```

```
    strSearch As String) As Integer
  ' Finds the last occurrence of the passed in string

  Dim intPos As Integer
  Dim intPrev As Integer

  On Error GoTo InstrRErr

  intPrev = intPos
  intPos = InStr(1, strMain, strSearch)

  Do While intPos > 0
    intPrev = intPos
    intPos = InStr(intPos + 1, strMain, strSearch)
  Loop
  InstrR = intPrev

  Exit Function

InstrRErr:
  Call LogErrors(Errors)
  gstrSource = mstrModuleName & "InstrR"
  On Error GoTo 0
  Err.Raise vbObjectError + errInstrRFailed, _
    gstrSource, _
    LoadResString(errInstrRFailed)

End Function

Public Function GetDefaultDir(lWspId As Long, WspParameters As
cArrParameters) As String

  Dim sDir As String
  sDir = SubstituteParameters( _
      gstrEnvVarSeparator & PARAM_DEFAULT_DIR &
gstrEnvVarSeparator, _
      lWspId, WspParameters:=WspParameters)
  MakePathValid (sDir & gstrFileSeparator & "a.txt")
  GetDefaultDir = GetShortName(sDir)
  If StringEmpty(GetDefaultDir) Then
    GetDefaultDir = App.Path
  End If

End Function

Public Sub AddArrayElement(ByRef arrNodes() As Object, _
    ByVal objToAdd As Object, _
    ByRef lngCount As Long)
  ' Adds the passed in object to the array

  On Error GoTo AddArrayElementErr

  ' Increase the array dimension and add the object to it
  ReDim Preserve arrNodes(lngCount)
  Set arrNodes(lngCount) = objToAdd
  lngCount = lngCount + 1

  Exit Sub

AddArrayElementErr:
  LogErrors Errors
  gstrSource = mstrModuleName & "AddArrayElement"
  On Error GoTo 0
  Err.Raise vbObjectError + errAddArrayElementFailed, _
      gstrSource, _
      LoadResString(errAddArrayElementFailed)

End Sub


Public Function CheckForNullField(rstRecords As Recordset, strFieldName As
String) As String

  ' Returns an empty string if a given field is null
  On Error GoTo CheckForNullFieldErr

  If IsNull(rstRecords.Fields(strFieldName)) Then
    CheckForNullField = gstrEmptyString
  Else
    CheckForNullField = rstRecords.Fields(strFieldName)
  End If
  Exit Function

CheckForNullFieldErr:
  Call LogErrors(Errors)
  On Error GoTo 0
  Err.Raise vbObjectError + errCheckForNullFieldFailed, _
      mstrModuleName & "CheckForNullField", _
      LoadResString(errCheckForNullFieldFailed)

End Function

Public Function ErrorOnNullField(rstRecords As Recordset, strFieldName As
String) As Variant

  ' If a given field is null, raises an error
  ' Else, returns the field value in a variant
  ' The calling function must convert the return value to the
  ' appropriate type
  On Error GoTo ErrorOnNullFieldErr
  gstrSource = mstrModuleName & "ErrorOnNullField"

  If IsNull(rstRecords.Fields(strFieldName)) Then
    On Error GoTo 0
    Err.Raise vbObjectError + errMandatoryFieldNull, _
      gstrSource, _
      strFieldName & mintDelimiter & LoadResString(errMandatoryFieldNull)
  Else
    ErrorOnNullField = rstRecords.Fields(strFieldName)
  End If
  Exit Function

ErrorOnNullFieldErr:
  Call LogErrors(Errors)
  On Error GoTo 0
  Err.Raise vbObjectError + errUnableToCheckNull, _
    gstrSource, _
    strFieldName & mintDelimiter & LoadResString(errUnableToCheckNull)

End Function
Public Function StringEmpty(strCheckString As String) As Boolean

  StringEmpty = (strCheckString = gstrEmptyString)

End Function
Public Function GetIteratorValue(cStepIterators As cRunColIt, _
    ByVal strItName As String)

  Dim lngIndex As Long
  Dim strValue As String

  On Error GoTo GetIteratorValueErr
  gstrSource = mstrModuleName & "GetIteratorValue"

  ' Find the iterator in the Iterators collection
  For lngIndex = 0 To cStepIterators.Count - 1
    If cStepIterators(lngIndex).IteratorName = strItName Then
      strValue = cStepIterators(lngIndex).Value
      Exit For
    End If
  Next lngIndex

  If lngIndex > cStepIterators.Count - 1 Then
    ' The iterator has not been defined for the branch
    ' Raise an error
    On Error GoTo 0
    Err.Raise vbObjectError + errParamNameInvalid, _
        gstrSource, _
```

```
            LoadResString(errParamNameInvalid)
    End If

    GetIteratorValue = strValue
    Exit Function

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

End Function
Public Function SubstituteParameters(ByVal strComString As String, _
        ByVal lngWorkspaceId As Long, _
        Optional StepIterators As cRunColIt = Nothing, _
        Optional WspParameters As cArrParameters = Nothing) As String
    ' This function substitutes all parameter names and
    ' environment variables in the passed in string with
    ' their values. It also substitutes the value for the
    ' iterators, if any.
    ' Since the syntax is to enclose parameter names and
    ' environment variables in "%", we check if a given
    ' variable is a parameter - if so, we substitute the
    ' parameter value - else we try to get the value from
    ' the environment

    Dim intPos As Integer
    Dim intEndPos As Integer
    Dim strEnvVariable As String
    Dim strValue As String
    Dim strCommand As String
    Dim cTempStr As cStringSM

    ' Initialize the return value of the function to the
    ' passed in command
    strCommand = strComString

    If WspParameters Is Nothing Then Set WspParameters = gcParameters

    Set cTempStr = New cStringSM

    intPos = InStr(strCommand, gstrEnvVarSeparator)
    Do While intPos <> 0
        If Mid(strCommand, intPos + 1, 1) = gstrEnvVarSeparator Then
            ' Wildcard character - to be substituted by a single % - later!
            intPos = intPos + 2
            If intPos > Len(strCommand) Then Exit Do
        Else
            ' Extract the environment variable from the passed
            ' in string
            intEndPos = InStr(intPos + 1, strCommand, gstrEnvVarSeparator)

            If intEndPos > 0 Then
                strEnvVariable = Mid(strCommand, intPos + 1, intEndPos - intPos - 1)
            Else
                On Error GoTo 0
                Err.Raise vbObjectError + errParamSeparatorMissing, _
                      gstrSource, _
                      LoadResString(errParamSeparatorMissing)
            End If
            strValue = gstrEmptyString

            ' Get the value of the variable and call a function
            ' to replace the variable with it's value
            strValue = GetValue(strEnvVariable, lngWorkspaceId, StepIterators,
WspParameters)
            ' The function raises an error if the variable is
            ' not found
            strCommand = cTempStr.ReplaceSubString(strCommand, _
                  gstrEnvVarSeparator & strEnvVariable & gstrEnvVarSeparator, _
                  strValue)
        End If

        intPos = InStr(intPos, strCommand, gstrEnvVarSeparator)
    Loop

    strCommand = cTempStr.ReplaceSubString(strCommand, _
          gstrEnvVarSeparator & gstrEnvVarSeparator, gstrEnvVarSeparator)

    Set cTempStr = Nothing
    SubstituteParameters = strCommand

End Function
Private Function GetValue(ByVal strParameter As String, _
        ByVal lngWorkspaceId As Long, _
        cStepIterators As cRunColIt, _
        WspParameters As cArrParameters) As String
    ' This function returns the value for the passed in
    ' parameter - it may be a workspace parameter, an
    ' environment variable or an iterator

    Dim intPos As Integer
    Dim intEndPos As Integer
    Dim strVariable As String
    Dim strValue As String
    Dim cParamRec As cParameter

    On Error GoTo GetValueErr

    ' Initialize the return value of the function to the
    ' empty
    strValue = gstrEmptyString

    intPos = InStr(strParameter, gstrEnvVarSeparator)
    If intPos > 0 Then
        ' Extract the variable from the passed in string
        intEndPos = InStr(intPos + 1, strParameter, gstrEnvVarSeparator)
        If intEndPos = 0 Then
            intEndPos = Len(strParameter)
        End If

        strVariable = Mid(strParameter, intPos + 1, intEndPos - intPos - 1)
    Else
        ' The separator charactor has not been passed in -
        ' try to find the value of the passed in parameter
        strVariable = strParameter
    End If

    If Not StringEmpty(strVariable) Then
        ' Check if this is the timestamp parameter first
        If strVariable = gstrTimeStamp Then
            strValue = Format$(Now, mstrFormatString, _
                  vbUseSystemDayOfWeek, vbUseSystem)
        Else
            ' Try to find a parameter for the workspace with
            ' the same name
            Set cParamRec = WspParameters.GetParameterValue(lngWorkspaceId, _
                  strVariable)
            If cParamRec Is Nothing Then
                If Not cStepIterators Is Nothing Then
                    ' If the string is not a parameter, then check
                    ' if it is an iterator
                    strValue = GetIteratorValue(cStepIterators, strVariable)
                End If

                If StringEmpty(strValue) Then
                    ' Neither - Check if it is an environment variable
                    strValue = Environ$(strVariable)
                    If StringEmpty(strValue) Then
                        On Error GoTo 0
                        WriteError errSubValuesFailed, _
                              OptArgs:="Invalid parameter: " & gstrSQ & strVariable &
gstrSQ
                        Err.Raise vbObjectError + errSubValuesFailed, _
```

```
                    mstrModuleName & "GetValue", _
                    LoadResString(errSubValuesFailed) & "Invalid parameter: "
& gstrSQ & strVariable & gstrSQ
                End If
            End If
        Else
            strValue = cParamRec.ParameterValue
        End If
    End If
End If

GetValue = strValue

Exit Function

GetValueErr:
If Err.Number = vbObjectError + errParamNameInvalid Then
    ' If the parameter has not been defined for the
    ' workspace then check if it is an environment
    ' variable
    Resume Next
End If

' Log the error code raised by Visual Basic
Call LogErrors(Errors)
gstrSource = mstrModuleName & "GetValue"
WriteError errSubValuesFailed, gstrSource, "Parameter: " & gstrSQ &
strVariable & gstrSQ
On Error GoTo 0
Err.Raise vbObjectError + errSubValuesFailed, _
        gstrSource, _
        LoadResString(errSubValuesFailed) & "Parameter: " & gstrSQ &
strVariable & gstrSQ

End Function
Public Function SQLFixup(strField As String) As String
' Returns a string that can be executed by SQL Server

Dim cMyStr As New cStringSM
Dim strTemp As String

On Error GoTo SQLFixupErr

strTemp = strField
SQLFixup = strTemp

' Single-quotes have to be replaced by two single-quotes,
' since a single-quote is the identifier delimiter
' character - call a procedure to do the replace
'   SQLFixup = cMyStr.ReplaceSubString(strTemp, gstrDQ, "\" & gstrDQ)

' Replace pipe characters with the corresponding chr function
' SQLFixup = cMyStr.ReplaceSubString(strTemp, gstrDQ, gstrDQ & gstrDQ)

Exit Function

SQLFixupErr:
gstrSource = mstrModuleName & "SQLFixup"
LogErrors Errors
On Error GoTo 0
Err.Raise vbObjectError + errMakeFieldValidFailed, _
        gstrSource, LoadResString(errMakeFieldValidFailed)

End Function
Public Function TranslateStepLabel(sLabel As String) As String
' Translates the passed in step label to a valid file name
' All characters in the label that are invalid for filenames (viz. \ / : * ? " < > |)
' and spaces are substituted with underscores - also ensure that the resulting
filename
' is not greater than 255 characters
Dim cTempStr As New cStringSM
TranslateStepLabel = cTempStr.ReplaceSubString(sLabel, gstrFileSeparator,
"_")
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, "/",
gstrUnderscore)
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, ":",
gstrUnderscore)
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, "*",
gstrUnderscore)
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, "?",
gstrUnderscore)
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, gstrDQ,
gstrUnderscore)
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, "<",
gstrUnderscore)
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, ">",
gstrUnderscore)
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, "|",
gstrUnderscore)
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel,
gstrBlank, gstrUnderscore)

' Commas are substituted with underscores since the command shell uses a
comma to
' delimit commands
TranslateStepLabel = cTempStr.ReplaceSubString(TranslateStepLabel, ",",
gstrUnderscore)

If Len(TranslateStepLabel) > MAX_PATH Then
    TranslateStepLabel = Mid(TranslateStepLabel, 1, MAX_PATH)
End If

End Function

Public Function TypeOfObject(ByVal objNode As Object) As Integer
' Determines the type of object that is passed in

On Error GoTo TypeOfObjectErr
gstrSource = mstrModuleName & "TypeOfObject"

Select Case TypeName(objNode)
    Case "cWorkspace"
        TypeOfObject = gintWorkspace

    Case "cParameter"
        TypeOfObject = gintParameter

    Case "cConnection"
        TypeOfObject = gintParameterConnect

    Case "cConnDtl"
        TypeOfObject = gintConnectionDtl

    Case "cGlobalStep"
        TypeOfObject = gintGlobalStep

    Case "cManager"
        TypeOfObject = gintManagerStep

    Case "cWorker"
        TypeOfObject = gintWorkerStep

    Case "cStep"
        ' If a step record is passed in, call a function
        ' to determine the type of step
        TypeOfObject = TypeOfStep(StepClass:=objNode)

    Case Else
        WriteError errTypeOfObjectFailed, gstrSource, _
            TypeName(objNode)
        On Error GoTo 0
        Err.Raise vbObjectError + errTypeOfObjectFailed, _
            gstrSource, _
            LoadResString(errTypeOfObjectFailed)
End Select

Exit Function
```

```
TypeOfObjectErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errTypeOfObjectFailed, _
         gstrSource, _
         LoadResString(errTypeOfObjectFailed)

End Function
```

ConnDtlCommon.bas

```
Attribute VB_Name = "ConnDtlCommon"
'  FILE:      ConnDtlCommon.bas
'            Microsoft TPC-H Kit Ver. 1.00
'            Copyright Microsoft, 1999
'            All Rights Reserved
'
'
'  PURPOSE:    Contains functionality common across StepMaster and
'            SMRunOnly, pertaining to connections
'            Specifically, functions to load connections in an array
'            and so on.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "ConnDtlCommon."

Public Sub LoadRSInConnDtlArray(rstConns As Recordset, cConns As
cConnDtls)

   Dim cNewConnDtl As cConnDtl

   On Error GoTo LoadRSInConnDtlArrayErr

   If rstConns.RecordCount = 0 Then
      Exit Sub
   End If

   rstConns.MoveFirst
   While Not rstConns.EOF

      Set cNewConnDtl = New cConnDtl

      ' Initialize ConnDtl values
      ' Call a procedure to raise an error if mandatory fields are null.
      cNewConnDtl.ConnNameId = ErrorOnNullField(rstConns,
FLD_ID_CONN_NAME)
      cNewConnDtl.WorkspaceId = ErrorOnNullField(rstConns,
FLD_ID_WORKSPACE)
      cNewConnDtl.ConnName = CStr(ErrorOnNullField(rstConns,
FLD_CONN_DTL_CONNECTION_NAME))
      cNewConnDtl.ConnectionString = CheckForNullField(rstConns,
FLD_CONN_DTL_CONNECTION_STRING)
      cNewConnDtl.ConnType = CheckForNullField(rstConns,
FLD_CONN_DTL_CONNECTION_TYPE)

      cConns.Load cNewConnDtl

      Set cNewConnDtl = Nothing
      rstConns.MoveNext
   Wend

   Exit Sub

LoadRSInConnDtlArrayErr:
   LogErrors Errors
   gstrSource = mstrModuleName & "LoadRSInConnDtlArray"
   On Error GoTo 0
   Err.Raise vbObjectError + errLoadRsInArrayFailed, gstrSource, _
         LoadResString(errLoadRsInArrayFailed)
End Sub
```

ConnectionCommon.bas

```
Attribute VB_Name = "ConnectionCommon"
'  FILE:      ConnnectionCommon.bas
'            Microsoft TPC-H Kit Ver. 1.00
'            Copyright Microsoft, 1999
'            All Rights Reserved
'
'
'  PURPOSE:    Contains functionality common across StepMaster and
'            SMRunOnly, pertaining to connection strings
'            Specifically, functions to load connections strings
'            in an array and so on.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "ConnectionCommon."

Public Sub LoadRecordsetInConnectionArray(rstConns As Recordset, cConns As
cConnections)

   Dim cNewConnection As cConnection

   On Error GoTo LoadRecordsetInConnectionArrayErr

   If rstConns.RecordCount = 0 Then
      Exit Sub
   End If

   rstConns.MoveFirst
   While Not rstConns.EOF

      Set cNewConnection = New cConnection

      ' Initialize Connection values
      ' Call a procedure to raise an error if mandatory fields are null.
      cNewConnection.ConnectionId = ErrorOnNullField(rstConns,
"connection_id")
      cNewConnection.WorkspaceId = CStr(ErrorOnNullField(rstConns,
FLD_ID_WORKSPACE))
      cNewConnection.ConnectionName = CStr(ErrorOnNullField(rstConns,
"connection_name"))
      cNewConnection.ConnectionValue = CheckForNullField(rstConns,
"connection_value")
      cNewConnection.Description = CheckForNullField(rstConns, "description")

      cNewConnection.NoCountDisplay = CheckForNullField(rstConns,
"no_count_display")
      cNewConnection.NoExecute = CheckForNullField(rstConns, "no_execute")
      cNewConnection.ParseQueryOnly = CheckForNullField(rstConns,
"parse_query_only")
      cNewConnection.QuotedIdentifiers = CheckForNullField(rstConns,
"ANSI_quoted_identifiers")
      cNewConnection.AnsiNulls = CheckForNullField(rstConns, "ANSI_nulls")
      cNewConnection.ShowQueryPlan = CheckForNullField(rstConns,
"show_query_plan")
      cNewConnection.ShowStatsTime = CheckForNullField(rstConns,
"show_stats_time")
      cNewConnection.ShowStatsIO = CheckForNullField(rstConns,
"show_stats_io")
      cNewConnection.ParseOdbcMsg = CheckForNullField(rstConns,
"parse_odbc_msg_prefixes")
      cNewConnection.RowCount = CheckForNullField(rstConns, "row_count")
      cNewConnection.TsqlBatchSeparator = CheckForNullField(rstConns,
"tsql_batch_separator")
```

```
        cNewConnection.QueryTimeOut = CheckForNullField(rstConns,
"query_time_out")
        cNewConnection.ServerLanguage = CheckForNullField(rstConns,
"server_language")
        cNewConnection.CharacterTranslation = CheckForNullField(rstConns,
"character_translation")
        cNewConnection.RegionalSettings = CheckForNullField(rstConns,
"regional_settings")

        cConns.Load cNewConnection

        Set cNewConnection = Nothing
        rstConns.MoveNext
    Wend

    Exit Sub

LoadRecordsetInConnectionArrayErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadRecordsetInConnectionArray"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, gstrSource, _
          LoadResString(errLoadRsInArrayFailed)
End Sub
```

cParameter.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cParameter"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cParameter.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of a parameter.
'              Contains functions to insert, update and delete
'              workspace_parameters records from the database.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit
Option Base 0

' Local variable(s) to hold property value(s)
Private mlngWorkspaceId As Long
Private mlngParameterId As Long
Private mstrParameterName As String
Private mstrParameterValue As String
Private mstrDescription As String
Private mintParameterType As Integer
Private mdbsStepMaster As Database
Private mintOperation As Operation
Private mlngPosition As Long

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cParameter."

' The cSequence class is used to generate unique parameter identifiers
Private mParameterSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM

' Parameter types
Public Enum ParameterType
    gintParameterGeneric = 0
    gintParameterConnect
    gintParameterApplication
    gintParameterBuiltIn
End Enum


Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the field names. When the parameter names are
    ' the same as the field names, parameters in the where
    ' clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[w_id]"
                prmParam.Value = mlngWorkspaceId

            Case "[p_id]"
                prmParam.Value = mlngParameterId

            Case "[p_name]"
                prmParam.Value = mstrParameterName

            Case "[p_value]"
                prmParam.Value = mstrParameterValue

            Case "[desc]"
                prmParam.Value = mstrDescription

            Case "[p_type]"
                prmParam.Value = mintParameterType

            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource, _
                      prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrModuleName &
"AssignParameters", _
                      LoadResString(errInvalidParameter)
        End Select
    Next prmParam

'    qyExec.Parameters("w_id").Value = mlngWorkspaceId
'    qyExec.Parameters("p_id").Value = mlngParameterId
'    qyExec.Parameters("p_name").Value = mstrParameterName
'    qyExec.Parameters("p_value").Value = mstrParameterValue
'

    Exit Sub

AssignParametersErr:

    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrSource, LoadResString(errAssignParametersFailed)

End Sub

Public Property Let Position(ByVal RHS As Long)

    mlngPosition = RHS

End Property
```

```
Public Property Get Position() As Long

   Position = mlngPosition

End Property

Public Function Clone() As cParameter

   ' Creates a copy of a given parameter

   Dim cCloneParam As cParameter

   On Error GoTo CloneErr
   mstrSource = mstrModuleName & "Clone"

   Set cCloneParam = New cParameter

   ' Copy all the parameter properties to the newly
   ' created parameter
   Set cCloneParam.NodeDB = mdbsStepMaster
   cCloneParam.WorkspaceId = mlngWorkspaceId
   cCloneParam.ParameterId = mlngParameterId
   cCloneParam.ParameterName = mstrParameterName
   cCloneParam.ParameterValue = mstrParameterValue
   cCloneParam.Description = mstrDescription
   cCloneParam.ParameterType = mintParameterType
   cCloneParam.IndOperation = mintOperation
   cCloneParam.Position = mlngPosition

   ' And set the return value to the newly created parameter
   Set Clone = cCloneParam
   Set cCloneParam = Nothing

   Exit Function

CloneErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Clone"
   On Error GoTo 0
   Err.Raise vbObjectError + errCloneFailed, _
         mstrSource, LoadResString(errCloneFailed)

End Function
Public Property Set NodeDB(vdata As Database)

   Set mdbsStepMaster = vdata

End Property
Public Property Get NodeDB() As Database

   Set NodeDB = mdbsStepMaster

End Property

Private Sub CheckDupParameterName()
   ' Check if the parameter name already exists in the workspace

   Dim rstParameter As Recordset
   Dim strSql As String
   Dim qy As DAO.QueryDef

   On Error GoTo CheckDupParameterNameErr
   mstrSource = mstrModuleName & "CheckDupParameterName"

   ' Create a recordset object to retrieve the count of all parameters
   ' for the workspace with the same name
   strSql = "Select count(*) as parameter_count " & _
      " from workspace_parameters " & _
      " where workspace_id = [w_id]" & _
      " and parameter_name = [p_name]" & _
      " and parameter_id <> [p_id]"

   Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strSql)
   Call AssignParameters(qy)

   Set rstParameter = qy.OpenRecordset(dbOpenForwardOnly)

   If rstParameter![parameter_count] > 0 Then
      rstParameter.Close
      qy.Close
      ShowError errDuplicateParameterName
      On Error GoTo 0
      Err.Raise vbObjectError + errDuplicateParameterName, _
         mstrSource, LoadResString(errDuplicateParameterName)
   End If

   rstParameter.Close
   qy.Close

   Exit Sub

CheckDupParameterNameErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "CheckDupParameterName"
   On Error GoTo 0
   Err.Raise vbObjectError + errCheckDupParameterNameFailed, _
      mstrSource, LoadResString(errCheckDupParameterNameFailed)

End Sub
Private Sub CheckDB()
   ' Check if the database object has been initialized

   If mdbsStepMaster Is Nothing Then
      On Error GoTo 0
      Err.Raise vbObjectError + errInvalidDB, _
         mstrModuleName & "CheckDB", LoadResString(errInvalidDB)
   End If

End Sub
Public Property Let ParameterValue(vdata As String)

   mstrParameterValue = vdata

End Property
Public Property Let Description(vdata As String)

   mstrDescription = vdata

End Property
Public Property Let ParameterType(vdata As ParameterType)

   mintParameterType = vdata

End Property

Public Property Let ParameterName(vdata As String)

   If vdata = gstrEmptyString Then

      ShowError errParameterNameMandatory
      On Error GoTo 0
      ' Propogate this error back to the caller
      Err.Raise vbObjectError + errParameterNameMandatory, _
         mstrSource, LoadResString(errParameterNameMandatory)
   Else
      mstrParameterName = vdata
   End If

End Property

Public Property Let ParameterId(vdata As Long)
   mlngParameterId = vdata
End Property

Public Property Let IndOperation(ByVal vdata As Operation)

   ' The valid operations are define in the cOperations
   ' class. Check if the operation is valid
```

```
    Select Case vdata
        Case QueryOp, InsertOp, UpdateOp, DeleteOp
            mintOperation = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errInvalidOperation, _
                mstrSource, LoadResString(errInvalidOperation)
    End Select

End Property
Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    On Error GoTo ValidateErr

    ' Check if the db object is valid
    Call CheckDB

    ' Call procedure to raise an error if the parameter name
    ' already exists in the workspace -
    ' if there are duplicates, we don't know what value for the
    ' parameter to use at runtime
    Call CheckDupParameterName

    Exit Sub

ValidateErr:

    mstrSource = mstrModuleName & "Validate"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errValidateFailed, _
        mstrSource, LoadResString(errValidateFailed)

End Sub
Public Property Let WorkspaceId(vdata As Long)

    mlngWorkspaceId = vdata

End Property

Public Sub Add()

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddErr

    ' Validate the record before trying to insert the record
    Call Validate

    ' Create a temporary querydef object
    strInsert = "insert into workspace_parameters " & _
            "( workspace_id, parameter_id, " & _
            " parameter_name, parameter_value, " & _
            " description, parameter_type ) " & _
            " values ( [w_id], [p_id], [p_name], [p_value], [desc], [p_type] ) "
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strInsert)

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

    qy.Execute dbFailOnError
    qy.Close

'    strInsert = "insert into workspace_parameters " & _
'            "( workspace_id, parameter_id, " & _
'            " parameter_name, parameter_value ) " & _
'            " values ( " & _
'            Str(mlngWorkspaceId) & ", " & Str(mlngParameterId) & _
'            ", " & mFieldValue.MakeStringFieldValid(mstrParameterName) & _
'            ", " & mFieldValue.MakeStringFieldValid(mstrParameterValue) & " ) "
'    mdbsStepMaster.Execute strInsert, dbFailOnError + dbSQLPassThrough

    Exit Sub

AddErr:

    mstrSource = mstrModuleName & "Add"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errParameterInsertFailed, _
        mstrSource, LoadResString(errParameterInsertFailed)

End Sub
Public Sub Delete()

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr

    ' Check if the db object is valid
    Call CheckDB

    strDelete = "delete from workspace_parameters " & _
            " where parameter_id = [p_id]"
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteParameterFailed, _
            mstrSource, _
            LoadResString(errDeleteParameterFailed)

End Sub

Public Sub Modify()

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo ModifyErr

    ' Validate the updated values before trying to modify the db
    Call Validate

    ' Create a temporary querydef object with the modify string
    strUpdate = "update workspace_parameters " & _
            " set workspace_id = [w_id], " & _
            "parameter_name = [p_name], " & _
            "parameter_value = [p_value], " & _
            "description = [desc], " & _
            "parameter_type = [p_type] " & _
            " where parameter_id = [p_id]"
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to assign the parameter values to the
    ' querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close
```

```
'    mdbsStepMaster.Execute strUpdate, dbFailOnError
'
    Exit Sub

ModifyErr:

    mstrSource = mstrModuleName & "Modify"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errParameterUpdateFailed, _
        mstrSource, LoadResString(errParameterUpdateFailed)

End Sub
Public Property Get ParameterName() As String

    ParameterName = mstrParameterName

End Property

Public Property Get ParameterId() As Long

    ParameterId = mlngParameterId

End Property
Public Property Get NextIdentifier() As Long

    Dim lngNextId As Long

    On Error GoTo NextIdentifierErr

    ' First check if the database object is valid
    Call CheckDB

    ' Retrieve the next identifier using the sequence class
    Set mParameterSeq = New cSequence
    Set mParameterSeq.IdDatabase = mdbsStepMaster
    mParameterSeq.IdentifierColumn = FLD_ID_PARAMETER
    lngNextId = mParameterSeq.Identifier
    Set mParameterSeq = Nothing

    NextIdentifier = lngNextId
    Exit Property

NextIdentifierErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "NextIdentifier"
    On Error GoTo 0
    Err.Raise vbObjectError + errIdGetFailed, _
        mstrSource, LoadResString(errIdGetFailed)

End Property
Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property

Public Property Get WorkspaceId() As Long

    WorkspaceId = mlngWorkspaceId

End Property

Public Property Get ParameterValue() As String

    ParameterValue = mstrParameterValue

End Property
Public Property Get Description() As String

    Description = mstrDescription

End Property
Public Property Get ParameterType() As ParameterType

    ParameterType = mintParameterType

End Property


Private Sub Class_Initialize()

    Set mFieldValue = New cStringSM

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp

End Sub

Private Sub Class_Terminate()

    Set mdbsStepMaster = Nothing
    Set mFieldValue = Nothing

End Sub
```

cRunColIt.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunColIt"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cRunColIt.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module implements a stack of Iterator nodes.
'              Ensures that only cRunItNode objects are stored in the stack.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cRunColIt."
Private mstrSource As String

Private mcIterators As cStack
Public Sub Clear()

    mcIterators.Clear

End Sub

Private Sub Class_Initialize()

    Set mcIterators = New cStack

End Sub

Private Sub Class_Terminate()

    Set mcIterators = Nothing

End Sub


Public Function Value(strItName As String) As String
```

```
    Dim lngIndex As Long

    For lngIndex = 0 To mcIterators.Count - 1
        If mcIterators(lngIndex).IteratorName = strItName Then
            Value = mcIterators(lngIndex).Value
            Exit For
        End If
    Next lngIndex

End Function

Public Property Get Item(ByVal Position As Long) As cRunItNode
Attribute Item.VB_UserMemId = 0

    Set Item = mcIterators(Position)

End Property


Public Function Count() As Long

    Count = mcIterators.Count

End Function

Public Function Pop() As cRunItNode

    Set Pop = mcIterators.Pop

End Function

Public Sub Push(objToPush As cRunItNode)

    Call mcIterators.Push(objToPush)

End Sub
```

cRunInst.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
  Persistable = 0  'NotPersistable
  DataBindingBehavior = 0  'vbNone
  DataSourceBehavior  = 0  'vbNone
  MTSTransactionMode  = 0  'NotAnMTSObject
END
Attribute VB_Name = "cRunInst"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cRunColIt.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     This module controls the run processing. It runs a branch
'               at a time and raises events when each step completes execution.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cRunInst."
Private mstrSource As String

' Local variable(s) to hold property value(s)
Private mstrRootKey As String
Public WspId As Long
Private mcParameters As cArrParameters
Private mcRunSteps As cArrSteps
Private mcRunConstraints As cArrConstraints
Public RunConnections As cConnections
Public RunConnDtls As cConnDtls
Private mcvntWspPreCons As Variant
Private mcvntWspPostCons As Variant
Private mcNavSteps As cStepTree

Private mcInstances As cInstances
Private mcFreeSteps As cVectorLng
Private mcFailures As cFailedSteps
Private mblnAsk As Boolean   ' Set to True when the a step with continuation
criteria=Ask fails
Private mblnAbort As Boolean ' Set to True when the run is aborted
Private msAbortDtls As String
Private mbarrFree() As Byte
Private WithEvents mcTermSteps As cTermSteps
Attribute mcTermSteps.VB_VarHelpID = -1
Public RunId As Long
Public CreateInputFiles As Boolean

Private Enum WspLogEvents
    mintRunStart
    mintRunComplete
    mintStepStart
    mintStepComplete
End Enum

Private mcWspLog As cFileSM

Private mstrCurBranchRoot As String
Private mcDummyRootInstance As cInstance

' Key for the dummy root instance - Should be a key that is invalid for an actual
step record
Private Const mstrDummyRootKey As String = "D"

' Public events to notify the calling function of the
' start and end time for each step
Public Event RunStart(dtmStartTime As Currency, strWspLog As String)
Public Event RunComplete(dtmEndTime As Currency)
Public Event StepStart(cStepRecord As cStep, dtmStartTime As Currency, _
        lngInstanceId As Long, lParentInstanceId As Long, sPath As String, _
        sIts As String, sItValue As String)
Public Event StepComplete(cStepRecord As cStep, dtmEndTime As Currency,
lngInstanceId As Long, lElapsed As Long)
Public Event ProcessStart(cStepRecord As cStep, strCommand As String, _
        dtmStartTime As Currency, lngInstanceId As Long, lParentInstanceId As
Long, _
        sItValue As String)
Public Event ProcessComplete(cStepRecord As cStep, dtmEndTime As Currency,
lngInstanceId As Long, lElapsed As Long)

' The class that will execute each step - we trap the events
' that are raised by it when a step starts/completes
' execution
Private WithEvents cExecStep1 As cRunStep
Attribute cExecStep1.VB_VarHelpID = -1
Private WithEvents cExecStep2 As cRunStep
Attribute cExecStep2.VB_VarHelpID = -1
Private WithEvents cExecStep3 As cRunStep
Attribute cExecStep3.VB_VarHelpID = -1
Private WithEvents cExecStep4 As cRunStep
Attribute cExecStep4.VB_VarHelpID = -1
Private WithEvents cExecStep5 As cRunStep
Attribute cExecStep5.VB_VarHelpID = -1
Private WithEvents cExecStep6 As cRunStep
Attribute cExecStep6.VB_VarHelpID = -1
Private WithEvents cExecStep7 As cRunStep
Attribute cExecStep7.VB_VarHelpID = -1
Private WithEvents cExecStep8 As cRunStep
Attribute cExecStep8.VB_VarHelpID = -1
Private WithEvents cExecStep9 As cRunStep
```

```
Attribute cExecStep9.VB_VarHelpID = -1

Private WithEvents cExecStep10 As cRunStep
Attribute cExecStep10.VB_VarHelpID = -1
Private WithEvents cExecStep11 As cRunStep
Attribute cExecStep11.VB_VarHelpID = -1
Private WithEvents cExecStep12 As cRunStep
Attribute cExecStep12.VB_VarHelpID = -1
Private WithEvents cExecStep13 As cRunStep
Attribute cExecStep13.VB_VarHelpID = -1
Private WithEvents cExecStep14 As cRunStep
Attribute cExecStep14.VB_VarHelpID = -1
Private WithEvents cExecStep15 As cRunStep
Attribute cExecStep15.VB_VarHelpID = -1
Private WithEvents cExecStep16 As cRunStep
Attribute cExecStep16.VB_VarHelpID = -1
Private WithEvents cExecStep17 As cRunStep
Attribute cExecStep17.VB_VarHelpID = -1
Private WithEvents cExecStep18 As cRunStep
Attribute cExecStep18.VB_VarHelpID = -1
Private WithEvents cExecStep19 As cRunStep
Attribute cExecStep19.VB_VarHelpID = -1

Private WithEvents cExecStep20 As cRunStep
Attribute cExecStep20.VB_VarHelpID = -1
Private WithEvents cExecStep21 As cRunStep
Attribute cExecStep21.VB_VarHelpID = -1
Private WithEvents cExecStep22 As cRunStep
Attribute cExecStep22.VB_VarHelpID = -1
Private WithEvents cExecStep23 As cRunStep
Attribute cExecStep23.VB_VarHelpID = -1
Private WithEvents cExecStep24 As cRunStep
Attribute cExecStep24.VB_VarHelpID = -1
Private WithEvents cExecStep25 As cRunStep
Attribute cExecStep25.VB_VarHelpID = -1
Private WithEvents cExecStep26 As cRunStep
Attribute cExecStep26.VB_VarHelpID = -1
Private WithEvents cExecStep27 As cRunStep
Attribute cExecStep27.VB_VarHelpID = -1
Private WithEvents cExecStep28 As cRunStep
Attribute cExecStep28.VB_VarHelpID = -1
Private WithEvents cExecStep29 As cRunStep
Attribute cExecStep29.VB_VarHelpID = -1

Private WithEvents cExecStep30 As cRunStep
Attribute cExecStep30.VB_VarHelpID = -1
Private WithEvents cExecStep31 As cRunStep
Attribute cExecStep31.VB_VarHelpID = -1
Private WithEvents cExecStep32 As cRunStep
Attribute cExecStep32.VB_VarHelpID = -1
Private WithEvents cExecStep33 As cRunStep
Attribute cExecStep33.VB_VarHelpID = -1
Private WithEvents cExecStep34 As cRunStep
Attribute cExecStep34.VB_VarHelpID = -1
Private WithEvents cExecStep35 As cRunStep
Attribute cExecStep35.VB_VarHelpID = -1
Private WithEvents cExecStep36 As cRunStep
Attribute cExecStep36.VB_VarHelpID = -1
Private WithEvents cExecStep37 As cRunStep
Attribute cExecStep37.VB_VarHelpID = -1
Private WithEvents cExecStep38 As cRunStep
Attribute cExecStep38.VB_VarHelpID = -1
Private WithEvents cExecStep39 As cRunStep
Attribute cExecStep39.VB_VarHelpID = -1

Private WithEvents cExecStep40 As cRunStep
Attribute cExecStep40.VB_VarHelpID = -1
Private WithEvents cExecStep41 As cRunStep
Attribute cExecStep41.VB_VarHelpID = -1
Private WithEvents cExecStep42 As cRunStep
Attribute cExecStep42.VB_VarHelpID = -1
Private WithEvents cExecStep43 As cRunStep
Attribute cExecStep43.VB_VarHelpID = -1
Private WithEvents cExecStep44 As cRunStep
Attribute cExecStep44.VB_VarHelpID = -1
Private WithEvents cExecStep45 As cRunStep
Attribute cExecStep45.VB_VarHelpID = -1
Private WithEvents cExecStep46 As cRunStep
Attribute cExecStep46.VB_VarHelpID = -1
Private WithEvents cExecStep47 As cRunStep
Attribute cExecStep47.VB_VarHelpID = -1
Private WithEvents cExecStep48 As cRunStep
Attribute cExecStep48.VB_VarHelpID = -1
Private WithEvents cExecStep49 As cRunStep
Attribute cExecStep49.VB_VarHelpID = -1

Private WithEvents cExecStep50 As cRunStep
Attribute cExecStep50.VB_VarHelpID = -1
Private WithEvents cExecStep51 As cRunStep
Attribute cExecStep51.VB_VarHelpID = -1
Private WithEvents cExecStep52 As cRunStep
Attribute cExecStep52.VB_VarHelpID = -1
Private WithEvents cExecStep53 As cRunStep
Attribute cExecStep53.VB_VarHelpID = -1
Private WithEvents cExecStep54 As cRunStep
Attribute cExecStep54.VB_VarHelpID = -1
Private WithEvents cExecStep55 As cRunStep
Attribute cExecStep55.VB_VarHelpID = -1
Private WithEvents cExecStep56 As cRunStep
Attribute cExecStep56.VB_VarHelpID = -1
Private WithEvents cExecStep57 As cRunStep
Attribute cExecStep57.VB_VarHelpID = -1
Private WithEvents cExecStep58 As cRunStep
Attribute cExecStep58.VB_VarHelpID = -1
Private WithEvents cExecStep59 As cRunStep
Attribute cExecStep59.VB_VarHelpID = -1

Private WithEvents cExecStep60 As cRunStep
Attribute cExecStep60.VB_VarHelpID = -1
Private WithEvents cExecStep61 As cRunStep
Attribute cExecStep61.VB_VarHelpID = -1
Private WithEvents cExecStep62 As cRunStep
Attribute cExecStep62.VB_VarHelpID = -1
Private WithEvents cExecStep63 As cRunStep
Attribute cExecStep63.VB_VarHelpID = -1
Private WithEvents cExecStep64 As cRunStep
Attribute cExecStep64.VB_VarHelpID = -1
Private WithEvents cExecStep65 As cRunStep
Attribute cExecStep65.VB_VarHelpID = -1
Private WithEvents cExecStep66 As cRunStep
Attribute cExecStep66.VB_VarHelpID = -1
Private WithEvents cExecStep67 As cRunStep
Attribute cExecStep67.VB_VarHelpID = -1
Private WithEvents cExecStep68 As cRunStep
Attribute cExecStep68.VB_VarHelpID = -1
Private WithEvents cExecStep69 As cRunStep
Attribute cExecStep69.VB_VarHelpID = -1

Private WithEvents cExecStep70 As cRunStep
Attribute cExecStep70.VB_VarHelpID = -1
Private WithEvents cExecStep71 As cRunStep
Attribute cExecStep71.VB_VarHelpID = -1
Private WithEvents cExecStep72 As cRunStep
Attribute cExecStep72.VB_VarHelpID = -1
Private WithEvents cExecStep73 As cRunStep
Attribute cExecStep73.VB_VarHelpID = -1
Private WithEvents cExecStep74 As cRunStep
Attribute cExecStep74.VB_VarHelpID = -1
Private WithEvents cExecStep75 As cRunStep
Attribute cExecStep75.VB_VarHelpID = -1
Private WithEvents cExecStep76 As cRunStep
Attribute cExecStep76.VB_VarHelpID = -1
Private WithEvents cExecStep77 As cRunStep
Attribute cExecStep77.VB_VarHelpID = -1
Private WithEvents cExecStep78 As cRunStep
Attribute cExecStep78.VB_VarHelpID = -1
Private WithEvents cExecStep79 As cRunStep
Attribute cExecStep79.VB_VarHelpID = -1
```

```
Private WithEvents cExecStep80 As cRunStep
Attribute cExecStep80.VB_VarHelpID = -1
Private WithEvents cExecStep81 As cRunStep
Attribute cExecStep81.VB_VarHelpID = -1
Private WithEvents cExecStep82 As cRunStep
Attribute cExecStep82.VB_VarHelpID = -1
Private WithEvents cExecStep83 As cRunStep
Attribute cExecStep83.VB_VarHelpID = -1
Private WithEvents cExecStep84 As cRunStep
Attribute cExecStep84.VB_VarHelpID = -1
Private WithEvents cExecStep85 As cRunStep
Attribute cExecStep85.VB_VarHelpID = -1
Private WithEvents cExecStep86 As cRunStep
Attribute cExecStep86.VB_VarHelpID = -1
Private WithEvents cExecStep87 As cRunStep
Attribute cExecStep87.VB_VarHelpID = -1
Private WithEvents cExecStep88 As cRunStep
Attribute cExecStep88.VB_VarHelpID = -1
Private WithEvents cExecStep89 As cRunStep
Attribute cExecStep89.VB_VarHelpID = -1

Private WithEvents cExecStep90 As cRunStep
Attribute cExecStep90.VB_VarHelpID = -1
Private WithEvents cExecStep91 As cRunStep
Attribute cExecStep91.VB_VarHelpID = -1
Private WithEvents cExecStep92 As cRunStep
Attribute cExecStep92.VB_VarHelpID = -1
Private WithEvents cExecStep93 As cRunStep
Attribute cExecStep93.VB_VarHelpID = -1
Private WithEvents cExecStep94 As cRunStep
Attribute cExecStep94.VB_VarHelpID = -1
Private WithEvents cExecStep95 As cRunStep
Attribute cExecStep95.VB_VarHelpID = -1
Private WithEvents cExecStep96 As cRunStep
Attribute cExecStep96.VB_VarHelpID = -1
Private WithEvents cExecStep97 As cRunStep
Attribute cExecStep97.VB_VarHelpID = -1
Private WithEvents cExecStep98 As cRunStep
Attribute cExecStep98.VB_VarHelpID = -1
Private WithEvents cExecStep99 As cRunStep
Attribute cExecStep99.VB_VarHelpID = -1

Private Const msIt As String = " Iterator: "
Private Const msItValue As String = " Value: "
Public Sub Abort()

   On Error GoTo AbortErr

   ' Make sure that we don't execute any more steps
   Call StopRun

   If cExecStep1 Is Nothing And cExecStep2 Is Nothing And cExecStep3 Is
Nothing And cExecStep4 Is Nothing And cExecStep5 Is Nothing And
cExecStep6 Is Nothing And cExecStep7 Is Nothing And cExecStep8 Is Nothing
And cExecStep9 Is Nothing And _
      cExecStep10 Is Nothing And cExecStep11 Is Nothing And cExecStep12 Is
Nothing And cExecStep13 Is Nothing And cExecStep14 Is Nothing And
cExecStep15 Is Nothing And cExecStep16 Is Nothing And cExecStep17 Is
Nothing And cExecStep18 Is Nothing And cExecStep19 Is Nothing And _
      cExecStep20 Is Nothing And cExecStep21 Is Nothing And cExecStep22 Is
Nothing And cExecStep23 Is Nothing And cExecStep24 Is Nothing And
cExecStep25 Is Nothing And cExecStep26 Is Nothing And cExecStep27 Is
Nothing And cExecStep28 Is Nothing And cExecStep29 Is Nothing And _
      cExecStep30 Is Nothing And cExecStep31 Is Nothing And cExecStep32 Is
Nothing And cExecStep33 Is Nothing And cExecStep34 Is Nothing And
cExecStep35 Is Nothing And cExecStep36 Is Nothing And cExecStep37 Is
Nothing And cExecStep38 Is Nothing And cExecStep39 Is Nothing And _
      cExecStep40 Is Nothing And cExecStep41 Is Nothing And cExecStep42 Is
Nothing And cExecStep43 Is Nothing And cExecStep44 Is Nothing And
cExecStep45 Is Nothing And cExecStep46 Is Nothing And cExecStep47 Is
Nothing And cExecStep48 Is Nothing And cExecStep49 Is Nothing And _
      cExecStep50 Is Nothing And cExecStep51 Is Nothing And cExecStep52 Is
Nothing And cExecStep53 Is Nothing And cExecStep54 Is Nothing And
cExecStep55 Is Nothing And cExecStep56 Is Nothing And cExecStep57 Is
Nothing And cExecStep58 Is Nothing And cExecStep59 Is Nothing And _
      cExecStep60 Is Nothing And cExecStep61 Is Nothing And cExecStep62 Is
Nothing And cExecStep63 Is Nothing And cExecStep64 Is Nothing And
cExecStep65 Is Nothing And cExecStep66 Is Nothing And cExecStep67 Is
Nothing And cExecStep68 Is Nothing And cExecStep69 Is Nothing And _
      cExecStep70 Is Nothing And cExecStep71 Is Nothing And cExecStep72 Is
Nothing And cExecStep73 Is Nothing And cExecStep74 Is Nothing And
cExecStep75 Is Nothing And cExecStep76 Is Nothing And cExecStep77 Is
Nothing And cExecStep78 Is Nothing And cExecStep79 Is Nothing And _
      cExecStep80 Is Nothing And cExecStep81 Is Nothing And cExecStep82 Is
Nothing And cExecStep83 Is Nothing And cExecStep84 Is Nothing And
cExecStep85 Is Nothing And cExecStep86 Is Nothing And cExecStep87 Is
Nothing And cExecStep88 Is Nothing And cExecStep89 Is Nothing And _
      cExecStep90 Is Nothing And cExecStep91 Is Nothing And cExecStep92 Is
Nothing And cExecStep93 Is Nothing And cExecStep94 Is Nothing And
cExecStep95 Is Nothing And cExecStep96 Is Nothing And cExecStep97 Is
Nothing And cExecStep98 Is Nothing And cExecStep99 Is Nothing Then
      ' Then...
      WriteToWspLog (mintRunComplete)
      RaiseEvent RunComplete(Determine64BitTime())
   Else
      ' Abort each of the steps that is currently executing.
      If Not cExecStep1 Is Nothing Then
         cExecStep1.Abort
      End If

      If Not cExecStep2 Is Nothing Then
         cExecStep2.Abort
      End If

      If Not cExecStep3 Is Nothing Then
         cExecStep3.Abort
      End If

      If Not cExecStep4 Is Nothing Then
         cExecStep4.Abort
      End If

      If Not cExecStep5 Is Nothing Then
         cExecStep5.Abort
      End If

      If Not cExecStep6 Is Nothing Then
         cExecStep6.Abort
      End If

      If Not cExecStep7 Is Nothing Then
         cExecStep7.Abort
      End If

      If Not cExecStep8 Is Nothing Then
         cExecStep8.Abort
      End If

      If Not cExecStep9 Is Nothing Then
         cExecStep9.Abort
      End If

      If Not cExecStep10 Is Nothing Then
         cExecStep10.Abort
      End If

      If Not cExecStep11 Is Nothing Then
         cExecStep11.Abort
      End If

      If Not cExecStep12 Is Nothing Then
         cExecStep12.Abort
      End If

      If Not cExecStep13 Is Nothing Then
         cExecStep13.Abort
      End If
```

```
If Not cExecStep14 Is Nothing Then
   cExecStep14.Abort
End If

If Not cExecStep15 Is Nothing Then
   cExecStep15.Abort
End If

If Not cExecStep16 Is Nothing Then
   cExecStep16.Abort
End If

If Not cExecStep17 Is Nothing Then
   cExecStep17.Abort
End If

If Not cExecStep18 Is Nothing Then
   cExecStep18.Abort
End If

If Not cExecStep19 Is Nothing Then
   cExecStep19.Abort
End If

If Not cExecStep20 Is Nothing Then
   cExecStep20.Abort
End If

If Not cExecStep21 Is Nothing Then
   cExecStep21.Abort
End If

If Not cExecStep22 Is Nothing Then
   cExecStep22.Abort
End If

If Not cExecStep23 Is Nothing Then
   cExecStep23.Abort
End If

If Not cExecStep24 Is Nothing Then
   cExecStep24.Abort
End If

If Not cExecStep25 Is Nothing Then
   cExecStep25.Abort
End If

If Not cExecStep26 Is Nothing Then
   cExecStep26.Abort
End If

If Not cExecStep27 Is Nothing Then
   cExecStep27.Abort
End If

If Not cExecStep28 Is Nothing Then
   cExecStep28.Abort
End If

If Not cExecStep29 Is Nothing Then
   cExecStep29.Abort
End If

' ============== 30 - 39 ==============
If Not cExecStep30 Is Nothing Then
   cExecStep30.Abort
End If

If Not cExecStep31 Is Nothing Then
   cExecStep31.Abort
End If

If Not cExecStep32 Is Nothing Then
   cExecStep32.Abort
End If

If Not cExecStep33 Is Nothing Then
   cExecStep33.Abort
End If

If Not cExecStep34 Is Nothing Then
   cExecStep34.Abort
End If

If Not cExecStep35 Is Nothing Then
   cExecStep35.Abort
End If

If Not cExecStep36 Is Nothing Then
   cExecStep36.Abort
End If

If Not cExecStep37 Is Nothing Then
   cExecStep37.Abort
End If

If Not cExecStep38 Is Nothing Then
   cExecStep38.Abort
End If

If Not cExecStep39 Is Nothing Then
   cExecStep39.Abort
End If

' ============== 40 - 49 ==============
If Not cExecStep40 Is Nothing Then
   cExecStep40.Abort
End If

If Not cExecStep41 Is Nothing Then
   cExecStep41.Abort
End If

If Not cExecStep42 Is Nothing Then
   cExecStep42.Abort
End If

If Not cExecStep43 Is Nothing Then
   cExecStep43.Abort
End If

If Not cExecStep44 Is Nothing Then
   cExecStep44.Abort
End If

If Not cExecStep45 Is Nothing Then
   cExecStep45.Abort
End If

If Not cExecStep46 Is Nothing Then
   cExecStep46.Abort
End If

If Not cExecStep47 Is Nothing Then
   cExecStep47.Abort
End If

If Not cExecStep48 Is Nothing Then
   cExecStep48.Abort
End If

If Not cExecStep49 Is Nothing Then
   cExecStep49.Abort
End If

' ============== 50 - 59 ==============
```

```
If Not cExecStep50 Is Nothing Then
   cExecStep50.Abort
End If

If Not cExecStep51 Is Nothing Then
   cExecStep51.Abort
End If

If Not cExecStep52 Is Nothing Then
   cExecStep52.Abort
End If

If Not cExecStep53 Is Nothing Then
   cExecStep53.Abort
End If

If Not cExecStep54 Is Nothing Then
   cExecStep54.Abort
End If

If Not cExecStep55 Is Nothing Then
   cExecStep55.Abort
End If

If Not cExecStep56 Is Nothing Then
   cExecStep56.Abort
End If

If Not cExecStep57 Is Nothing Then
   cExecStep57.Abort
End If

If Not cExecStep58 Is Nothing Then
   cExecStep58.Abort
End If

If Not cExecStep59 Is Nothing Then
   cExecStep59.Abort
End If

' ============== 60 - 69 ==============
If Not cExecStep60 Is Nothing Then
   cExecStep60.Abort
End If

If Not cExecStep61 Is Nothing Then
   cExecStep61.Abort
End If

If Not cExecStep62 Is Nothing Then
   cExecStep62.Abort
End If

If Not cExecStep63 Is Nothing Then
   cExecStep63.Abort
End If

If Not cExecStep64 Is Nothing Then
   cExecStep64.Abort
End If

If Not cExecStep65 Is Nothing Then
   cExecStep65.Abort
End If

If Not cExecStep66 Is Nothing Then
   cExecStep66.Abort
End If

If Not cExecStep67 Is Nothing Then
   cExecStep67.Abort
End If

If Not cExecStep68 Is Nothing Then
   cExecStep68.Abort
End If

If Not cExecStep69 Is Nothing Then
   cExecStep69.Abort
End If

' ============== 70 - 79 ==============
If Not cExecStep70 Is Nothing Then
   cExecStep70.Abort
End If

If Not cExecStep71 Is Nothing Then
   cExecStep71.Abort
End If

If Not cExecStep72 Is Nothing Then
   cExecStep72.Abort
End If

If Not cExecStep73 Is Nothing Then
   cExecStep73.Abort
End If

If Not cExecStep74 Is Nothing Then
   cExecStep74.Abort
End If

If Not cExecStep75 Is Nothing Then
   cExecStep75.Abort
End If

If Not cExecStep76 Is Nothing Then
   cExecStep76.Abort
End If

If Not cExecStep77 Is Nothing Then
   cExecStep77.Abort
End If

If Not cExecStep78 Is Nothing Then
   cExecStep78.Abort
End If

If Not cExecStep79 Is Nothing Then
   cExecStep79.Abort
End If

' ============== 80 - 89 ==============
If Not cExecStep80 Is Nothing Then
   cExecStep80.Abort
End If

If Not cExecStep81 Is Nothing Then
   cExecStep81.Abort
End If

If Not cExecStep82 Is Nothing Then
   cExecStep82.Abort
End If

If Not cExecStep83 Is Nothing Then
   cExecStep83.Abort
End If

If Not cExecStep84 Is Nothing Then
   cExecStep84.Abort
End If

If Not cExecStep85 Is Nothing Then
   cExecStep85.Abort
End If

If Not cExecStep86 Is Nothing Then
```

```
        cExecStep86.Abort
      End If

      If Not cExecStep87 Is Nothing Then
        cExecStep87.Abort
      End If

      If Not cExecStep88 Is Nothing Then
        cExecStep88.Abort
      End If

      If Not cExecStep89 Is Nothing Then
        cExecStep89.Abort
      End If

      ' ============== 90 - 99 ==============
      If Not cExecStep90 Is Nothing Then
        cExecStep90.Abort
      End If

      If Not cExecStep91 Is Nothing Then
        cExecStep91.Abort
      End If

      If Not cExecStep92 Is Nothing Then
        cExecStep92.Abort
      End If

      If Not cExecStep93 Is Nothing Then
        cExecStep93.Abort
      End If

      If Not cExecStep94 Is Nothing Then
        cExecStep94.Abort
      End If

      If Not cExecStep95 Is Nothing Then
        cExecStep95.Abort
      End If

      If Not cExecStep96 Is Nothing Then
        cExecStep96.Abort
      End If

      If Not cExecStep97 Is Nothing Then
        cExecStep97.Abort
      End If

      If Not cExecStep98 Is Nothing Then
        cExecStep98.Abort
      End If

      If Not cExecStep99 Is Nothing Then
        cExecStep99.Abort
      End If

    End If

    Exit Sub

AbortErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    ShowError errAbortFailed
    ' Try to abort the remaining steps, if any
    Resume Next

End Sub
Public Sub AbortSiblings(cTermInstance As cInstance)

    On Error GoTo AbortSiblingsErr

    ' Abort each of the steps that is currently executing.
    If Not cExecStep1 Is Nothing Then
      If cExecStep1.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep1.Abort
      End If
    End If

    If Not cExecStep2 Is Nothing Then
      If cExecStep2.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep2.Abort
      End If
    End If

    If Not cExecStep3 Is Nothing Then
      If cExecStep3.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep3.Abort
      End If
    End If

    If Not cExecStep4 Is Nothing Then
      If cExecStep4.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep4.Abort
      End If
    End If

    If Not cExecStep5 Is Nothing Then
      If cExecStep5.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep5.Abort
      End If
    End If

    If Not cExecStep6 Is Nothing Then
      If cExecStep6.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep6.Abort
      End If
    End If

    If Not cExecStep7 Is Nothing Then
      If cExecStep7.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep7.Abort
      End If
    End If

    If Not cExecStep8 Is Nothing Then
      If cExecStep8.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep8.Abort
      End If
    End If

    If Not cExecStep9 Is Nothing Then
      If cExecStep9.ExecuteStep.ParentStepId = cTermInstance.Step.ParentStepId
Then
        cExecStep9.Abort
      End If
    End If

    If Not cExecStep10 Is Nothing Then
      If cExecStep10.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep10.Abort
      End If
    End If

    If Not cExecStep11 Is Nothing Then
      If cExecStep11.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep11.Abort
      End If
```

```
    End If

    If Not cExecStep12 Is Nothing Then
        If cExecStep12.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep12.Abort
        End If
    End If

    If Not cExecStep13 Is Nothing Then
        If cExecStep13.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep13.Abort
        End If
    End If

    If Not cExecStep14 Is Nothing Then
        If cExecStep14.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep14.Abort
        End If
    End If

    If Not cExecStep15 Is Nothing Then
        If cExecStep15.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep15.Abort
        End If
    End If

    If Not cExecStep16 Is Nothing Then
        If cExecStep16.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep16.Abort
        End If
    End If

    If Not cExecStep17 Is Nothing Then
        If cExecStep17.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep17.Abort
        End If
    End If

    If Not cExecStep18 Is Nothing Then
        If cExecStep18.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep18.Abort
        End If
    End If

    If Not cExecStep19 Is Nothing Then
        If cExecStep19.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep19.Abort
        End If
    End If

    If Not cExecStep20 Is Nothing Then
        If cExecStep20.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep20.Abort
        End If
    End If

    If Not cExecStep21 Is Nothing Then
        If cExecStep21.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep21.Abort
        End If
    End If

    If Not cExecStep22 Is Nothing Then
        If cExecStep22.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep22.Abort
        End If
    End If

    If Not cExecStep23 Is Nothing Then
        If cExecStep23.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep23.Abort
        End If
    End If

    If Not cExecStep24 Is Nothing Then
        If cExecStep24.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep24.Abort
        End If
    End If

    If Not cExecStep25 Is Nothing Then
        If cExecStep25.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep25.Abort
        End If
    End If

    If Not cExecStep26 Is Nothing Then
        If cExecStep26.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep26.Abort
        End If
    End If

    If Not cExecStep27 Is Nothing Then
        If cExecStep27.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep27.Abort
        End If
    End If

    If Not cExecStep28 Is Nothing Then
        If cExecStep28.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep28.Abort
        End If
    End If

    If Not cExecStep29 Is Nothing Then
        If cExecStep29.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep29.Abort
        End If
    End If

    ' ============ 30 =================
    If Not cExecStep30 Is Nothing Then
        If cExecStep30.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep30.Abort
        End If
    End If

    If Not cExecStep31 Is Nothing Then
        If cExecStep31.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep31.Abort
        End If
    End If

    If Not cExecStep32 Is Nothing Then
        If cExecStep32.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep32.Abort
```

```
        End If
    End If

    If Not cExecStep33 Is Nothing Then
        If cExecStep33.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep33.Abort
        End If
    End If

    If Not cExecStep34 Is Nothing Then
        If cExecStep34.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep34.Abort
        End If
    End If

    If Not cExecStep35 Is Nothing Then
        If cExecStep35.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep35.Abort
        End If
    End If

    If Not cExecStep36 Is Nothing Then
        If cExecStep36.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep36.Abort
        End If
    End If

    If Not cExecStep37 Is Nothing Then
        If cExecStep37.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep37.Abort
        End If
    End If

    If Not cExecStep38 Is Nothing Then
        If cExecStep38.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep38.Abort
        End If
    End If

    If Not cExecStep39 Is Nothing Then
        If cExecStep39.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep39.Abort
        End If
    End If

    ' ============ 40 =================
    If Not cExecStep40 Is Nothing Then
        If cExecStep40.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep40.Abort
        End If
    End If

    If Not cExecStep41 Is Nothing Then
        If cExecStep41.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep41.Abort
        End If
    End If

    If Not cExecStep42 Is Nothing Then
        If cExecStep42.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep42.Abort
        End If
    End If

    If Not cExecStep43 Is Nothing Then
        If cExecStep43.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep43.Abort
        End If
    End If

    If Not cExecStep44 Is Nothing Then
        If cExecStep44.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep44.Abort
        End If
    End If

    If Not cExecStep45 Is Nothing Then
        If cExecStep45.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep45.Abort
        End If
    End If

    If Not cExecStep46 Is Nothing Then
        If cExecStep46.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep46.Abort
        End If
    End If

    If Not cExecStep47 Is Nothing Then
        If cExecStep47.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep47.Abort
        End If
    End If

    If Not cExecStep48 Is Nothing Then
        If cExecStep48.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep48.Abort
        End If
    End If

    If Not cExecStep49 Is Nothing Then
        If cExecStep49.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep49.Abort
        End If
    End If

    ' ============ 50 =================
    If Not cExecStep50 Is Nothing Then
        If cExecStep50.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep50.Abort
        End If
    End If

    If Not cExecStep51 Is Nothing Then
        If cExecStep51.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep51.Abort
        End If
    End If

    If Not cExecStep52 Is Nothing Then
        If cExecStep52.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep52.Abort
        End If
    End If

    If Not cExecStep53 Is Nothing Then
        If cExecStep53.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
```

```
        cExecStep53.Abort
      End If
    End If

    If Not cExecStep54 Is Nothing Then
      If cExecStep54.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep54.Abort
      End If
    End If

    If Not cExecStep55 Is Nothing Then
      If cExecStep55.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep55.Abort
      End If
    End If

    If Not cExecStep56 Is Nothing Then
      If cExecStep56.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep56.Abort
      End If
    End If

    If Not cExecStep57 Is Nothing Then
      If cExecStep57.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep57.Abort
      End If
    End If

    If Not cExecStep58 Is Nothing Then
      If cExecStep58.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep58.Abort
      End If
    End If

    If Not cExecStep59 Is Nothing Then
      If cExecStep59.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep59.Abort
      End If
    End If

    ' ============ 60 =================
    If Not cExecStep60 Is Nothing Then
      If cExecStep60.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep60.Abort
      End If
    End If

    If Not cExecStep61 Is Nothing Then
      If cExecStep61.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep61.Abort
      End If
    End If

    If Not cExecStep62 Is Nothing Then
      If cExecStep62.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep62.Abort
      End If
    End If

    If Not cExecStep63 Is Nothing Then
      If cExecStep63.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep63.Abort
      End If
    End If

    If Not cExecStep64 Is Nothing Then
      If cExecStep64.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep64.Abort
      End If
    End If

    If Not cExecStep65 Is Nothing Then
      If cExecStep65.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep65.Abort
      End If
    End If

    If Not cExecStep66 Is Nothing Then
      If cExecStep66.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep66.Abort
      End If
    End If

    If Not cExecStep67 Is Nothing Then
      If cExecStep67.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep67.Abort
      End If
    End If

    If Not cExecStep68 Is Nothing Then
      If cExecStep68.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep68.Abort
      End If
    End If

    If Not cExecStep69 Is Nothing Then
      If cExecStep69.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep69.Abort
      End If
    End If

    ' ============ 70 =================
    If Not cExecStep70 Is Nothing Then
      If cExecStep70.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep70.Abort
      End If
    End If

    If Not cExecStep71 Is Nothing Then
      If cExecStep71.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep71.Abort
      End If
    End If

    If Not cExecStep72 Is Nothing Then
      If cExecStep72.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep72.Abort
      End If
    End If

    If Not cExecStep73 Is Nothing Then
      If cExecStep73.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
        cExecStep73.Abort
      End If
    End If

    If Not cExecStep74 Is Nothing Then
```

```
        If cExecStep74.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep74.Abort
        End If
    End If

    If Not cExecStep75 Is Nothing Then
        If cExecStep75.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep75.Abort
        End If
    End If

    If Not cExecStep76 Is Nothing Then
        If cExecStep76.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep76.Abort
        End If
    End If

    If Not cExecStep77 Is Nothing Then
        If cExecStep77.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep77.Abort
        End If
    End If

    If Not cExecStep78 Is Nothing Then
        If cExecStep78.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep78.Abort
        End If
    End If

    If Not cExecStep79 Is Nothing Then
        If cExecStep79.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep79.Abort
        End If
    End If

    ' ============ 80 =================
    If Not cExecStep80 Is Nothing Then
        If cExecStep80.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep80.Abort
        End If
    End If

    If Not cExecStep81 Is Nothing Then
        If cExecStep81.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep81.Abort
        End If
    End If

    If Not cExecStep82 Is Nothing Then
        If cExecStep82.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep82.Abort
        End If
    End If

    If Not cExecStep83 Is Nothing Then
        If cExecStep83.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep83.Abort
        End If
    End If

    If Not cExecStep84 Is Nothing Then
        If cExecStep84.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep84.Abort
        End If
    End If

    If Not cExecStep85 Is Nothing Then
        If cExecStep85.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep85.Abort
        End If
    End If

    If Not cExecStep86 Is Nothing Then
        If cExecStep86.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep86.Abort
        End If
    End If

    If Not cExecStep87 Is Nothing Then
        If cExecStep87.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep87.Abort
        End If
    End If

    If Not cExecStep88 Is Nothing Then
        If cExecStep88.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep88.Abort
        End If
    End If

    If Not cExecStep89 Is Nothing Then
        If cExecStep89.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep89.Abort
        End If
    End If

    ' ============ 90 =================
    If Not cExecStep90 Is Nothing Then
        If cExecStep90.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep90.Abort
        End If
    End If

    If Not cExecStep91 Is Nothing Then
        If cExecStep91.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep91.Abort
        End If
    End If

    If Not cExecStep92 Is Nothing Then
        If cExecStep92.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep92.Abort
        End If
    End If

    If Not cExecStep93 Is Nothing Then
        If cExecStep93.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep93.Abort
        End If
    End If

    If Not cExecStep94 Is Nothing Then
        If cExecStep94.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
            cExecStep94.Abort
        End If
    End If
```

```
   If Not cExecStep95 Is Nothing Then
      If cExecStep95.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
         cExecStep95.Abort
      End If
   End If

   If Not cExecStep96 Is Nothing Then
      If cExecStep96.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
         cExecStep96.Abort
      End If
   End If

   If Not cExecStep97 Is Nothing Then
      If cExecStep97.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
         cExecStep97.Abort
      End If
   End If

   If Not cExecStep98 Is Nothing Then
      If cExecStep98.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
         cExecStep98.Abort
      End If
   End If

   If Not cExecStep99 Is Nothing Then
      If cExecStep99.ExecuteStep.ParentStepId =
cTermInstance.Step.ParentStepId Then
         cExecStep99.Abort
      End If
   End If

   Exit Sub

AbortSiblingsErr:
   Call LogErrors(Errors)
   On Error GoTo 0
   ShowError errAbortFailed
   ' Try to abort the remaining steps, if any
   Resume Next

End Sub
Private Sub ExecutionFailed(cTermStep As cRunStep)
   ' Called when execution of a step fails for any reason - ensure that execution
   ' continues

   On Error GoTo ExecutionFailedErr

   Call AddFreeProcess(cTermStep.Index)

   Call RunBranch(mstrCurBranchRoot)

   Exit Sub

ExecutionFailedErr:
   ' Log the error code raised by Visual Basic - do not raise an error here!
   Call LogErrors(Errors)

End Sub
Private Sub FreeExecStep(lngIndex As Long)
   ' Frees an instance of a cExecuteSM object depending on the index
   On Error GoTo FreeExecStepErr

   Select Case lngIndex + 1
      Case 1
         Set cExecStep1 = Nothing
      Case 2
         Set cExecStep2 = Nothing
      Case 3
         Set cExecStep3 = Nothing
      Case 4
         Set cExecStep4 = Nothing
      Case 5
         Set cExecStep5 = Nothing
      Case 6
         Set cExecStep6 = Nothing
      Case 7
         Set cExecStep7 = Nothing
      Case 8
         Set cExecStep8 = Nothing
      Case 9
         Set cExecStep9 = Nothing
      Case 10
         Set cExecStep10 = Nothing
      Case 11
         Set cExecStep11 = Nothing
      Case 12
         Set cExecStep12 = Nothing
      Case 13
         Set cExecStep13 = Nothing
      Case 14
         Set cExecStep14 = Nothing
      Case 15
         Set cExecStep15 = Nothing
      Case 16
         Set cExecStep16 = Nothing
      Case 17
         Set cExecStep17 = Nothing
      Case 18
         Set cExecStep18 = Nothing
      Case 19
         Set cExecStep19 = Nothing
      Case 20
         Set cExecStep20 = Nothing
      Case 21
         Set cExecStep21 = Nothing
      Case 22
         Set cExecStep22 = Nothing
      Case 23
         Set cExecStep23 = Nothing
      Case 24
         Set cExecStep24 = Nothing
      Case 25
         Set cExecStep25 = Nothing
      Case 26
         Set cExecStep26 = Nothing
      Case 27
         Set cExecStep27 = Nothing
      Case 28
         Set cExecStep28 = Nothing
      Case 29
         Set cExecStep29 = Nothing
      Case 30
         Set cExecStep30 = Nothing
      Case 31
         Set cExecStep31 = Nothing
      Case 32
         Set cExecStep32 = Nothing
      Case 33
         Set cExecStep33 = Nothing
      Case 34
         Set cExecStep34 = Nothing
      Case 35
         Set cExecStep35 = Nothing
      Case 36
         Set cExecStep36 = Nothing
      Case 37
         Set cExecStep37 = Nothing
      Case 38
         Set cExecStep38 = Nothing
      Case 39
         Set cExecStep39 = Nothing
      Case 40
         Set cExecStep40 = Nothing
      Case 41
```

```
    Set cExecStep41 = Nothing
Case 42
    Set cExecStep42 = Nothing
Case 43
    Set cExecStep43 = Nothing
Case 44
    Set cExecStep44 = Nothing
Case 45
    Set cExecStep45 = Nothing
Case 46
    Set cExecStep46 = Nothing
Case 47
    Set cExecStep47 = Nothing
Case 48
    Set cExecStep48 = Nothing
Case 49
    Set cExecStep49 = Nothing
Case 50
    Set cExecStep50 = Nothing
Case 51
    Set cExecStep51 = Nothing
Case 52
    Set cExecStep52 = Nothing
Case 53
    Set cExecStep53 = Nothing
Case 54
    Set cExecStep54 = Nothing
Case 55
    Set cExecStep55 = Nothing
Case 56
    Set cExecStep56 = Nothing
Case 57
    Set cExecStep57 = Nothing
Case 58
    Set cExecStep58 = Nothing
Case 59
    Set cExecStep59 = Nothing
Case 60
    Set cExecStep60 = Nothing
Case 61
    Set cExecStep61 = Nothing
Case 62
    Set cExecStep62 = Nothing
Case 63
    Set cExecStep63 = Nothing
Case 64
    Set cExecStep64 = Nothing
Case 65
    Set cExecStep65 = Nothing
Case 66
    Set cExecStep66 = Nothing
Case 67
    Set cExecStep67 = Nothing
Case 68
    Set cExecStep68 = Nothing
Case 69
    Set cExecStep69 = Nothing
Case 70
    Set cExecStep70 = Nothing
Case 71
    Set cExecStep71 = Nothing
Case 72
    Set cExecStep72 = Nothing
Case 73
    Set cExecStep73 = Nothing
Case 74
    Set cExecStep74 = Nothing
Case 75
    Set cExecStep75 = Nothing
Case 76
    Set cExecStep76 = Nothing
Case 77
    Set cExecStep77 = Nothing
Case 78
    Set cExecStep78 = Nothing
  Case 79
    Set cExecStep79 = Nothing
  Case 80
    Set cExecStep80 = Nothing
  Case 81
    Set cExecStep81 = Nothing
  Case 82
    Set cExecStep82 = Nothing
  Case 83
    Set cExecStep83 = Nothing
  Case 84
    Set cExecStep84 = Nothing
  Case 85
    Set cExecStep85 = Nothing
  Case 86
    Set cExecStep86 = Nothing
  Case 87
    Set cExecStep87 = Nothing
  Case 88
    Set cExecStep88 = Nothing
  Case 89
    Set cExecStep89 = Nothing
  Case 90
    Set cExecStep90 = Nothing
  Case 91
    Set cExecStep91 = Nothing
  Case 92
    Set cExecStep92 = Nothing
  Case 93
    Set cExecStep93 = Nothing
  Case 94
    Set cExecStep94 = Nothing
  Case 95
    Set cExecStep95 = Nothing
  Case 96
    Set cExecStep96 = Nothing
  Case 97
    Set cExecStep97 = Nothing
  Case 98
    Set cExecStep98 = Nothing
  Case 99
    Set cExecStep99 = Nothing
  Case Else
    BugAssert False, "FreeExecStep: Invalid index value!"
  End Select

  Exit Sub

FreeExecStepErr:
  ' Log the error code raised by Visual Basic
  Call LogErrors(Errors)

End Sub
Private Sub ProcessAskFailures()
  ' This procedure is called when a step with a continuation criteria = Ask has
failed.
  ' Wait for all running processes to complete before displaying an
Abort/Retry/Fail
  ' message to the user. We process every Ask step that has failed and use a
simple
  ' algorithm to determine what to do next.
  ' 1. An abort response to any failure results in an immediate abort of the run
  ' 2. A continue means the run continues - this failure is popped off the failure
list.
  ' 3. A retry means that the execution details for the instance are cleared and the
  '    step is re-executed.
  Dim lIndex As Long
  Dim cStepRec As cStep
  Dim cNextInst As cInstance
  Dim cFailureRec As cFailedStep

  On Error GoTo ProcessAskFailuresErr
```

```
   ' Display a popup message for all steps that have failed with a continuation
   ' criteria of Ask
   For lIndex = mcFailures.Count - 1 To 0 Step -1

      Set cFailureRec = mcFailures(lIndex)

      If cFailureRec.ContCriteria = gintOnFailureAsk Then
         Set cStepRec = mcRunSteps.QueryStep(cFailureRec.StepId)
         ' Ask the user whether to abort/retry/continue
         #If RUN_ONLY Then
            cFailureRec.AskResponse = ShowMessageBox(0, _
                  "Step '" & GetStepNodeText(cStepRec) & "' failed. " & _
                  "Select Abort to abort run and Ignore to continue. " & _
                  "Select Retry to re-execute the failed step.", _
                  "Step Failure", _
                  MB_ABORTRETRYIGNORE + MB_APPLMODAL +
MB_ICONEXCLAMATION)
         #Else
            cFailureRec.AskResponse = ShowMessageBox(frmRunning.hWnd, _
                  "Step '" & GetStepNodeText(cStepRec) & "' failed. " & _
                  "Select Abort to abort run and Ignore to continue. " & _
                  "Select Retry to re-execute the failed step.", _
                  "Step Failure", _
                  MB_ABORTRETRYIGNORE + MB_APPLMODAL +
MB_ICONEXCLAMATION)
         #End If

         ' Process an abort response immediately
         If cFailureRec.AskResponse = IDABORT Then
            mblnAbort = True
            Set cNextInst = mcInstances.QueryInstance(cFailureRec.InstanceId)
            Call RunPendingSiblings(cNextInst, cFailureRec.EndTime)
            Exit For
         End If
      End If

   Next lIndex

   ' Process all failed steps for which we have Ignore and Retry responses.
   If Not mblnAbort Then
      ' Navigate in reverse order since we'll be deleting items from the collection
      For lIndex = mcFailures.Count - 1 To 0 Step -1
         If mcFailures(lIndex).ContCriteria = gintOnFailureAsk Then
            mblnAsk = False
            Set cFailureRec = mcFailures.Delete(lIndex)

            Select Case cFailureRec.AskResponse
               Case IDABORT
                  BugAssert True

               Case IDRETRY
                  ' Delete all instances for the failed step and re-try
                  ' Returns a parent instance reference
                  Set cNextInst = ProcessRetryStep(cFailureRec)
                  Call RunPendingStepInBranch(mstrCurBranchRoot, cNextInst)

               Case IDIGNORE
                  Set cNextInst =
mcInstances.QueryInstance(cFailureRec.InstanceId)
                  Call RunPendingSiblings(cNextInst, cFailureRec.EndTime)

            End Select
         End If
      Next lIndex
   End If

   Exit Sub

ProcessAskFailuresErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   Err.Raise vbObjectError + errExecuteBranchFailed, mstrModuleName, _
         LoadResString(errExecuteBranchFailed)

End Sub
Private Function ProcessRetryStep(cFailureRec As cFailedStep) As cInstance
   ' This procedure is called when a step with a continuation criteria = Ask has
failed
   ' and the user wants to re-execute the step.
   ' We delete all existing instances for the step and reset the iterator, if
   ' any on the parent instance - this way we ensure that the step will be executed
   ' in the next pass.
   Dim lIndex As Long
   Dim cParentInstance As cInstance
   Dim cSubStepRec As cSubStep
   Dim cStepRec As cStep

   On Error GoTo ProcessRetryStepErr

   ' Navigate in reverse order since we'll be deleting items from the collection
   For lIndex = mcInstances.Count - 1 To 0 Step -1

      If mcInstances(lIndex).Step.StepId = cFailureRec.StepId Then
         Set cParentInstance =
mcInstances.QueryInstance(mcInstances(lIndex).ParentInstanceId)
         Set cSubStepRec = cParentInstance.QuerySubStep(cFailureRec.StepId)
         Set cStepRec = mcRunSteps.QueryStep(cFailureRec.StepId)

         ' Decrement the child count on the parent instance and reset the
         ' step iterators on the sub-step record, if any -
         ' all the iterations of the step will be re-executed.
         cParentInstance.ChildDeleted cFailureRec.StepId
         cParentInstance.AllComplete = False
         cParentInstance.AllStarted = False

         cSubStepRec.InitializeIt cStepRec, mcParameters

         ' Now delete the current instance
         Set ProcessRetryStep = mcInstances.Delete(lIndex)
      End If
   Next lIndex

   Exit Function

ProcessRetryStepErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   Err.Raise vbObjectError + errExecuteBranchFailed, mstrModuleName, _
         LoadResString(errExecuteBranchFailed)

End Function


Private Sub RunNextStep(ByVal dtmCompleteTime As Currency, ByVal
lngIndex As Long, _
      ByVal InstanceId As Long, ByVal ExecutionStatus As InstanceStatus)
   ' Checks if there are any steps remaining to be
   ' executed in the current branch. If so, it executes
   ' the step.
   Dim cTermInstance As cInstance
   Dim cFailure As cFailedStep

   On Error GoTo RunNextStepErr

   BugMessage "RunNextStep: cExecStep" & CStr(lngIndex + 1) & " has
completed."

   Call mcTermSteps.Delete
   Call FreeExecStep(lngIndex)

   ' Call a procedure to add the freed up object to the list
   Call AddFreeProcess(lngIndex)

   Set cTermInstance = mcInstances.QueryInstance(InstanceId)
   cTermInstance.Status = ExecutionStatus

   If ExecutionStatus = gintFailed Then
```

```
        If cTermInstance.Step.ContinuationCriteria = gintOnFailureAbortSiblings
Then
            Call AbortSiblings(cTermInstance)
        End If

        If Not mcFailures.StepFailed(cTermInstance.Step.StepId) Then
            Set cFailure = New cFailedStep
            cFailure.InstanceId = cTermInstance.InstanceId
            cFailure.StepId = cTermInstance.Step.StepId
            cFailure.ParentStepId = cTermInstance.Step.ParentStepId
            cFailure.ContCriteria = cTermInstance.Step.ContinuationCriteria
            cFailure.EndTime = dtmCompleteTime
            mcFailures.Add cFailure
            Set cFailure = Nothing
        End If
    End If

    If ExecutionStatus = gintFailed And cTermInstance.Step.ContinuationCriteria
= gintOnFailureAbort Then
        If StringEmpty(msAbortDtls) Then
            ' Initialize the abort message
            msAbortDtls = "Step '" & GetStepNodeText(cTermInstance.Step) & "'
failed. " & _
                    "Aborting execution. Please check the error file for details."
        End If
        Call Abort
    ElseIf ExecutionStatus = gintFailed And
cTermInstance.Step.ContinuationCriteria = gintOnFailureAsk Then
        mblnAsk = True

        ' If the step failed due to a Cancel operation (Abort), abort the run
        If mblnAbort Then
            Call RunPendingSiblings(cTermInstance, dtmCompleteTime)
        End If
    Else
        Call RunPendingSiblings(cTermInstance, dtmCompleteTime)
    End If

    If mblnAbort Then
        If Not AnyStepRunning(mcFreeSteps, mbarrFree) And Not
StringEmpty(msAbortDtls) Then
            ' Display an error only if the abort is due to a failure
            ' We had to abort since a step failed - since no other steps are currently
            ' running, we can display a message to the user saying that we had to abort
            #If RUN_ONLY Then
                Call ShowMessageBox(0, msAbortDtls, "Run Aborted", _
                        MB_APPLMODAL + MB_OK + MB_ICONEXCLAMATION)
            #Else
                Call ShowMessageBox(frmRunning.hWnd, msAbortDtls, "Run
Aborted", _
                        MB_APPLMODAL + MB_OK + MB_ICONEXCLAMATION)
            #End If
'            MsgBox msAbortDtls, vbOKOnly, "Run Aborted"
        End If
    ElseIf mblnAsk Then
        If Not AnyStepRunning(mcFreeSteps, mbarrFree) Then
            ' Ask the user whether to abort/retry/ignore failed steps
            Call ProcessAskFailures
        End If
    End If

    Exit Sub

RunNextStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    WriteError errExecuteBranchFailed, mstrSource
    Call ResetForm(lngIndex)

End Sub
Public Sub StopRun()

    ' Setting the Abort flag to True will ensure that we
    ' don't execute any more steps
    mblnAbort = True

End Sub

Private Sub CreateDummyInstance(strRootKey As String)

    Dim cNewInstance As cInstance
    Dim cSubStepDtls  As cStep
    Dim lngSubStepId As Long

    On Error GoTo CreateDummyInstanceErr

    ' Create a new instance of the step
    ' initialize substeps for the step
    Set cNewInstance = New cInstance

    ' There can be multiple iterations of the top level nodes
    ' running at the same time, but only one branch at any
    ' time - so enforce a degree of parallelism of 1 on this
    ' node!
    Set cNewInstance.Step = New cStep
    cNewInstance.DegreeParallelism = 1
    cNewInstance.Key = mstrDummyRootKey

    cNewInstance.InstanceId = NewInstanceId
    cNewInstance.ParentInstanceId = 0

    lngSubStepId = MakeIdentifierValid(strRootKey)

    Set cSubStepDtls = mcRunSteps.QueryStep(lngSubStepId)
    If cSubStepDtls.EnabledFlag Then
        ' Create a child node for the step corresponding to
        ' the root node of the branch being currently executed,
        ' only if it has been enabled
        Call cNewInstance.CreateSubStep(cSubStepDtls, mcParameters)
    End If

    mcInstances.Add cNewInstance
    Set cNewInstance.Iterators = DetermineIterators(cNewInstance)

    ' Set a reference to the newly created dummy instance
    Set mcDummyRootInstance = cNewInstance

    Set cNewInstance = Nothing

    Exit Sub

CreateDummyInstanceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "CreateDummyInstance"
    Err.Raise vbObjectError + errCreateInstanceFailed, _
            mstrSource, LoadResString(errCreateInstanceFailed)

End Sub
Private Function CreateInstance(cExecStep As cStep, _
        cParentInstance As cInstance) As cInstance
    ' Creates a new instance of the passed in step. Returns
    ' a reference to the newly created instance object.

    Dim cNewInstance As cInstance
    Dim nodChild As cStep
    Dim lngSubStepId As Long

    On Error GoTo CreateInstanceErr

    ' Create a new instance of the step
    ' initialize substeps for the step
    Set cNewInstance = New cInstance
    Set cNewInstance.Step = cExecStep
    cNewInstance.Key = MakeKeyValid(cExecStep.StepId, cExecStep.StepType)
    cNewInstance.ParentInstanceId = cParentInstance.InstanceId
    cNewInstance.InstanceId = NewInstanceId
```

```
    ' Validate the degree of parallelism field before assigning it to the instance -
    ' (the parameter value might have been set to an invalid value at runtime)
    Call ValidateParallelism(cExecStep.DegreeParallelism, _
            cExecStep.WorkspaceId, ParamsInWsp:=mcParameters)
    cNewInstance.DegreeParallelism =
SubstituteParameters(cExecStep.DegreeParallelism, _
            cExecStep.WorkspaceId, WspParameters:=mcParameters)

    If mcNavSteps.HasChild(StepKey:=cNewInstance.Key) Then
        Set nodChild = mcNavSteps.ChildStep(StepKey:=cNewInstance.Key)
        Do
            If nodChild.EnabledFlag Then
                ' Create nodes for all it's substeps only
                ' if the substeps have been enabled
                Call cNewInstance.CreateSubStep(nodChild, mcParameters)
            End If

            Set nodChild = mcNavSteps.NextStep(StepId:=nodChild.StepId)
        Loop While (Not nodChild Is Nothing)
    End If

    mcInstances.Add cNewInstance
    Set cNewInstance.Iterators = DetermineIterators(cNewInstance)

    ' Increment the number of executing steps on the parent
    cParentInstance.ChildExecuted (cExecStep.StepId)

    Set CreateInstance = cNewInstance

    Exit Function

CreateInstanceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "CreateInstance"
    Err.Raise vbObjectError + errCreateInstanceFailed, _
            mstrSource, LoadResString(errCreateInstanceFailed)

End Function
Private Function DetermineIterators(cInstanceRec As cInstance) As cRunColIt
    ' Returns a collection of all the iterator values for this
    ' instance - since an iterator that is defined at a
    ' particular level can be used in all it's substeps, we
    ' need to navigate the step tree all the way to the root

    Dim cRunIts As cRunColIt
    Dim cRunIt As cRunItNode
    Dim cStepIt As cIterator
    Dim cParentInst As cInstance
    Dim cSubStepRec As cSubStep
    Dim cSubStepDtls As cStep
    Dim lngSubStepId As Long
    Dim lngIndex As Long

    On Error GoTo DetermineIteratorsErr

    Set cRunIts = New cRunColIt

    If cInstanceRec.ParentInstanceId > 0 Then
        ' The last iterator for an instance of a step is stored
        ' on it's parent! So navigate up before beginning the
        ' search for iterator values.
        Set cParentInst = mcInstances.QueryInstance(cInstanceRec.ParentInstanceId)

        ' Get the sub-step record for the current step
        ' on it's parent's instance!
        lngSubStepId = cInstanceRec.Step.StepId
        Set cSubStepRec = cParentInst.QuerySubStep(lngSubStepId)
        Set cSubStepDtls = mcRunSteps.QueryStep(lngSubStepId)

        ' And determine the next iteration value for the
        ' substep in this instance
        Set cStepIt = cSubStepRec.NewIteration(cSubStepDtls)

        If Not cStepIt Is Nothing Then
            ' Add the iterator details to the collection since
            ' an iterator has been defined for the step
            Set cRunIt = New cRunItNode
            cRunIt.IteratorName = cSubStepDtls.IteratorName
            cRunIt.Value = SubstituteParameters(cStepIt.Value,
cSubStepDtls.WorkspaceId, WspParameters:=mcParameters)
            cRunIt.StepId = cSubStepRec.StepId
            cRunIts.Push cRunIt
        End If

        ' Since the parent instance has all the iterators upto
        ' that level, read them and push them on to the stack for
        ' this instance
        For lngIndex = 0 To cParentInst.Iterators.Count - 1
            Set cRunIt = cParentInst.Iterators(lngIndex)
            cRunIts.Push cRunIt
        Next lngIndex
    End If

    Set DetermineIterators = cRunIts

    Exit Function

DetermineIteratorsErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "DetermineIterators"
    Err.Raise vbObjectError + errExecInstanceFailed, _
            mstrSource, LoadResString(errExecInstanceFailed)

End Function
Private Function DetermineConstraints(cInstanceRec As cInstance, _
        intConsType As ConstraintType) As Variant
    ' Returns a collection of all the constraints for this
    ' instance of the passed in type - all the constraints defined
    ' for the manager are executed first, followed by those defined
    ' for the step. If a step has an iterator defined for it, each
    ' constraint is executed only once.

    Dim cParentInst As cInstance
    Dim cTempInst As cInstance
    Dim vntConstraints As Variant
    Dim vntTempCons As Variant
    Dim cColConstraints() As Variant
    Dim lngConsCount As Long

    On Error GoTo DetermineConstraintsErr

    Set cTempInst = cInstanceRec
    lngConsCount = 0

    ' Go all the way to the root
    Do
        If cTempInst.ParentInstanceId > 0 Then
            Set cParentInst = mcInstances.QueryInstance(cTempInst.ParentInstanceId)
        Else
            Set cParentInst = Nothing
        End If

        ' Check if the step has an iterator defined for it
        If cTempInst.ValidForIteration(cParentInst, intConsType) Then
            vntTempCons = mcRunConstraints.ConstraintsForStep( _
                    cTempInst.Step.StepId, cTempInst.Step.VersionNo, _
                    intConsType, blnSort:=True, _
                    blnGlobal:=False, blnGlobalConstraintsOnly:=False)

            If Not IsEmpty(vntTempCons) Then
                ReDim Preserve cColConstraints(lngConsCount)
                cColConstraints(lngConsCount) = vntTempCons
                lngConsCount = lngConsCount + 1
            End If
```

```
        End If

        Set cTempInst = cParentInst

    Loop While Not cTempInst Is Nothing

    If lngConsCount > 0 Then
        vntTempCons = OrderConstraints(cColConstraints, intConsType)
    End If

    DetermineConstraints = vntTempCons

    Exit Function

DetermineConstraintsErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "DetermineConstraints"
    Err.Raise vbObjectError + errExecInstanceFailed, _
            mstrSource, LoadResString(errExecInstanceFailed)

End Function
Private Function GetInstanceToExecute(cParentNode As cInstance, _
        cSubStepRec As cSubStep, _
        cSubStepDtls As cStep) As cInstance

    Dim cSubStepInst As cInstance

    On Error GoTo GetInstanceToExecuteErr

    BugAssert Not (cParentNode Is Nothing Or _
            cSubStepRec Is Nothing Or _
            cSubStepDtls Is Nothing), _
            "GetInstanceToExecute: Input invalid"

    ' Check if it has iterators
    If cSubStepDtls.IteratorCount = 0 Then
        ' Check if the step has been executed
        If cSubStepRec.TasksRunning = 0 And cSubStepRec.TasksComplete = 0
And _
                Not mcInstances.CompletedInstanceExists(cParentNode.InstanceId,
cSubStepDtls) Then
            ' The sub-step hasn't been executed yet.
            ' Create an instance for it and exit
            Set cSubStepInst = CreateInstance(cSubStepDtls, cParentNode)
        Else
            Set cSubStepInst = Nothing
        End If
    Else
        ' Check if there are pending iterations for the sub-step
        If Not cSubStepRec.NextIteration(cSubStepDtls) Is Nothing Then
            ' Pending iterations exist - create an instance for the sub-step and exit
            Set cSubStepInst = CreateInstance(cSubStepDtls, cParentNode)
        Else
            ' No more iterations - continue with the next substep
            Set cSubStepInst = Nothing
        End If
    End If

    Set GetInstanceToExecute = cSubStepInst
    Exit Function

GetInstanceToExecuteErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "GetInstanceToExecute"
    Err.Raise vbObjectError + errNavInstancesFailed, _
            mstrSource, LoadResString(errNavInstancesFailed)

End Function
Public Function InstancesForStep(lngStepId As Long, ByRef StepStatus As
InstanceStatus) As cInstances
    ' Returns an array of all the instances for a step
    Dim lngIndex As Long
    Dim cTempInst As cInstance
    Dim cStepInstances As cInstances
    Dim cStepRec As cStep

    On Error GoTo InstancesForStepErr

    Set cStepInstances = New cInstances

    For lngIndex = 0 To mcInstances.Count - 1
        Set cTempInst = mcInstances(lngIndex)

        If cTempInst.Step.StepId = lngStepId Then
            cStepInstances.Add cTempInst
        End If
    Next lngIndex

    If cStepInstances.Count = 0 Then
        Set cStepRec = mcRunSteps.QueryStep(lngStepId)
        If Not mcFailures.ExecuteSubStep(cStepRec.ParentStepId) Then
            StepStatus = gintAborted
        End If
        Set cStepRec = Nothing
    End If

    ' Set the return value of the function to the array of
    ' constraints that has been built above
    Set InstancesForStep = cStepInstances

    Set cStepInstances = Nothing
    Exit Function

InstancesForStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "InstancesForStep"
    Err.Raise vbObjectError + errNavInstancesFailed, mstrSource, _
            LoadResString(errNavInstancesFailed)

End Function
Private Sub RemoveFreeProcess(lngRunningProcess As Long)
    ' Removes the passed in element from the collection of
    ' free objects

    ' Confirm that the last element in the array is the one
    ' we need to delete
    If mcFreeSteps(mcFreeSteps.Count - 1) = lngRunningProcess Then
        mcFreeSteps.Delete Position:=mcFreeSteps.Count - 1
    Else
        ' Ask the class to find the element and delete it
        mcFreeSteps.Delete Item:=lngRunningProcess
    End If

End Sub
Private Sub AddFreeProcess(lngTerminatedProcess As Long)
    ' Adds the passed in element to the collection of
    ' free objects

    mcFreeSteps.Add lngTerminatedProcess

End Sub

Private Sub ResetForm(Optional ByVal lngIndex As Long)

    Dim lngTemp As Long

    On Error GoTo ResetFormErr

    ' Check if there are any running instances to wait for
    If mcFreeSteps.Count <> glngNumConcurrentProcesses Then
```

```
      For lngTemp = 0 To mcFreeSteps.Count - 1
         If mcFreeSteps(lngTemp) = lngIndex Then
            Exit For
         End If
      Next lngTemp

      If lngTemp <= mcFreeSteps.Count - 1 Then
         ' This process that just completed did not exist in the list of
         ' free processes
         Call AddFreeProcess(lngIndex)
      End If

      If Not AnyStepRunning(mcFreeSteps, mbarrFree) Then
         WriteToWspLog (mintRunComplete)
         ' All steps are complete
         RaiseEvent RunComplete(Determine64BitTime())
      End If
   Else
      WriteToWspLog (mintRunComplete)
      RaiseEvent RunComplete(Determine64BitTime())
   End If

   Exit Sub

ResetFormErr:

End Sub
Private Function NewInstanceId() As Long
   ' Will return new instance id's - uses a static counter
   ' that it increments each time
   Static lngInstance As Long

   lngInstance = lngInstance + 1
   NewInstanceId = lngInstance

End Function

Private Function RunPendingStepInBranch(strCurBranchRoot As String, _
      Optional cExecInstance As cInstance = Nothing) As cInstance
   ' Runs a worker step in the branch being executed, if
   ' there are any pending execution
   ' This function is also called when a step has just completed
   ' execution - in which case the terminated instance is
   ' passed in as the optional parameter. When that happens,
   ' we first try to execute the siblings of the terminated
   ' step if any are pending execution.
   ' If the terminated instance has not been passed in, we
   ' start with the dummy root instance and navigate down,
   ' trying to find a pending worker step.

   Dim cExecSubStep As cStep
   Dim cParentInstance As cInstance
   Dim cNextInst As cInstance

   On Error GoTo RunPendingStepInBranchErr

   If Not cExecInstance Is Nothing Then
      ' Called when an instance has terminated
      ' When a worker step terminates, then we need to
      ' decremement the number of running steps on it's
      ' manager
      Set cParentInstance = _
            mcInstances.QueryInstance(cExecInstance.ParentInstanceId)

   Else
      If StringEmpty(strCurBranchRoot) Or mcDummyRootInstance Is Nothing
Then
         ' Run complete - event raised by Run method
         Set RunPendingStepInBranch = Nothing
         Exit Function
      End If

      ' If there are no pending steps on the root instance,
      ' then there are no steps within the branch that need
      ' to be executed
      If mcDummyRootInstance.AllComplete Or
mcDummyRootInstance.AllStarted Then
         Set RunPendingStepInBranch = Nothing
         Exit Function
      End If

      Set cParentInstance = mcDummyRootInstance
   End If

   Do
      Set cNextInst = GetSubStepToExecute(cParentInstance)
      If cNextInst Is Nothing Then
         ' There are no steps within the branch that can
         ' be executed - If we are at the dummy instance,
         ' this branch has completed executing
         If cParentInstance.Key = mstrDummyRootKey Then
            Set cNextInst = Nothing
            Exit Do
         Else
            ' Go to the parent instance and try to find
            ' some other sibling is pending execution
            Set cNextInst =
mcInstances.QueryInstance(cParentInstance.ParentInstanceId)

            If cParentInstance.SubSteps.Count = 0 Then
               cNextInst.ChildTerminated cParentInstance.Step.StepId
            End If
         End If
      End If

      BugAssert Not cNextInst Is Nothing
      Set cParentInstance = cNextInst

   Loop While cNextInst.Step.StepType <> gintWorkerStep

   If Not cNextInst Is Nothing Then
      Call ExecuteStep(cNextInst)
   End If

   Set RunPendingStepInBranch = cNextInst

   Exit Function

RunPendingStepInBranchErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errNavInstancesFailed, _
         mstrModuleName & "RunPendingStepInBranch",
LoadResString(errNavInstancesFailed)

End Function
Private Function RunPendingSibling(cTermInstance As cInstance, _
      dtmCompleteTime As Currency) As cInstance
   ' This process is called when a step terminates. Tries to
   ' run a sibling of the terminated step, if one is pending
   ' execution.

   Dim cParentInstance As cInstance
   Dim cNextInst As cInstance

   On Error GoTo RunPendingSiblingErr

   If StringEmpty(mstrCurBranchRoot) Or mcDummyRootInstance Is Nothing
Then
      ' Run complete - event raised by Run method
      Set RunPendingSibling = Nothing
      Exit Function
   End If

   BugAssert cTermInstance.ParentInstanceId > 0, "Orphaned instance in array!"
```

```
   ' When a worker step terminates, then we need to
   ' decremement the number of running steps on it's
   ' manager
   Set cParentInstance =
mcInstances.QueryInstance(cTermInstance.ParentInstanceId)

   ' Decrement the number of running processes on the
   ' parent by 1
   Call cParentInstance.ChildTerminated(cTermInstance.Step.StepId)

   ' The first step that terminates has to be a worker
   ' If it is complete, update the completed steps on the
   ' parent by 1.
   Call cParentInstance.ChildCompleted(cTermInstance.Step.StepId)
   cParentInstance.AllStarted = False

   Do
      Set cNextInst = GetSubStepToExecute(cParentInstance, dtmCompleteTime)
      If cNextInst Is Nothing Then
         If cParentInstance.Key = mstrDummyRootKey Then
            Set cNextInst = Nothing
            Exit Do
         Else
            ' Go to the parent instance and try to find
            ' some other sibling is pending execution
            Set cNextInst =
mcInstances.QueryInstance(cParentInstance.ParentInstanceId)
            If cParentInstance.IsRunning Then
               cNextInst.AllStarted = True
            Else
               ' No more sub-steps to execute
               Call cNextInst.ChildCompleted(cParentInstance.Step.StepId)
               Call cNextInst.ChildTerminated(cParentInstance.Step.StepId)
               cNextInst.AllStarted = False
            End If
         End If
      End If

      BugAssert Not cNextInst Is Nothing
      Set cParentInstance = cNextInst

   Loop While cNextInst.Step.StepType <> gintWorkerStep

   If Not cNextInst Is Nothing Then
      Call ExecuteStep(cNextInst)
   End If

   Set RunPendingSibling = cNextInst

   Exit Function

RunPendingSiblingErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   mstrSource = mstrModuleName & "RunPendingSibling"
   Err.Raise vbObjectError + errNavInstancesFailed, mstrSource, _
      LoadResString(errNavInstancesFailed)

End Function
Private Sub RunPendingSiblings(cTermInstance As cInstance, _
      dtmCompleteTime As Currency)
   ' This process is called when a step terminates. Tries to
   ' run siblings of the terminated step, if they are pending
   ' execution.

   Dim cExecInst As cInstance

   On Error GoTo RunPendingSiblingsErr
   BugMessage "In RunPendingSiblings"

   ' Call a procedure to run the sibling of the terminated
   ' step, if any. This procedure will also update the
   ' number of complete/running tasks on the manager steps.
   Set cExecInst = RunPendingSibling(cTermInstance, dtmCompleteTime)

   If Not cExecInst Is Nothing Then
      Do
         ' Execute any other pending steps in the branch.
         ' The step that has just terminated might be
         ' the last one that was executing in a sub-branch.
         ' That would mean that we can execute another
         ' sub-branch that might involve more than 1 step.
         ' Pass the just executed step as a parameter.
         Set cExecInst = RunPendingStepInBranch(mstrCurBranchRoot,
cExecInst)
      Loop While Not cExecInst Is Nothing
   Else
      If Not mcDummyRootInstance.IsRunning Then
         ' All steps have been executed in the branch - run
         ' a new branch
         Call RunNewBranch
      Else
         ' There are no more steps to execute in the current
         ' branch but we have running processes.
      End If
   End If

   Exit Sub

RunPendingSiblingsErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   mstrSource = mstrModuleName & "RunPendingSiblings"
   Err.Raise vbObjectError + errNavInstancesFailed, _
         mstrSource, LoadResString(errNavInstancesFailed)

End Sub

Private Sub NoSubStepsToExecute(cMgrInstance As cInstance, Optional
dtmCompleteTime As Currency = gdtmEmpty)
   ' Called when we cannot find any more substeps to run for
   ' manager step - set the allcomplete or allstarted
   ' properties to true

   If cMgrInstance.IsRunning() Then
      cMgrInstance.AllStarted = True
   Else
      cMgrInstance.AllComplete = True
      If dtmCompleteTime <> gdtmEmpty Then
         ' Update the end time on the manager step
         Call TimeCompleteUpdateForStep(cMgrInstance, dtmCompleteTime)
      End If
   End If

End Sub

Private Function GetSubStepToExecute(cParentNode As cInstance, _
      Optional dtmCompleteTime As Currency = 0) As cInstance
   ' Returns the child of the passed in node that is to be
   ' executed next. Checks if we are in the middle of an instance
   ' being executed in which case it returns the pending
   ' instance. Creates a new instance if there are pending
   ' instances for a sub-step.

   Dim lngIndex As Long
   Dim cSubStepRec As cSubStep
   Dim cSubStepDtls As cStep
   Dim cSubStepInst As cInstance

   On Error GoTo GetSubStepToExecuteErr

   ' There are a number of cases that need to be accounted
   ' for here.
   ' 1. While traversing through all enabled nodes for the
   ' first time - instance records may not exist for the
   ' substeps.
```

```
    ' 2. Instance records exist, and there are processes
    ' that need to be executed for a sub-step
    ' 3. There are no more processes that need to be currently
    ' executed (till a process completes)
    ' 4. There are no more processes that need to be executed
    ' (All substeps have completed execution)

    ' This is the only point where we check the Abort flag -
    ' since this is the heart of the navigation routine that
    ' selects processes to execute. Also, when a step terminates
    ' selection of the next process goes through here.
    If mblnAbort Then
        Set GetSubStepToExecute = Nothing
        cParentNode.Status = gintAborted
        Exit Function
    End If

    If mblnAsk Then
        Set GetSubStepToExecute = Nothing
        Exit Function
    End If

    If Not mcFailures.ExecuteSubStep(cParentNode.Step.StepId) Then
        Set GetSubStepToExecute = Nothing
        cParentNode.Status = gintAborted
        Exit Function
    End If

    ' First check if there are pending steps for the parent!
    If cParentNode.IsPending Then
        ' Loop through all the sub-steps for the parent node
        For lngIndex = 0 To cParentNode.SubSteps.Count - 1
            Set cSubStepRec = cParentNode.SubSteps(lngIndex)
            Set cSubStepDtls = mcRunSteps.QueryStep(cSubStepRec.StepId)
            If Not mcInstances.InstanceAborted(cSubStepRec) Then
                ' Check if the sub-step is a worker
                If cSubStepDtls.StepType = gintWorkerStep Then
                    ' Find/create an instance to execute
                    Set cSubStepInst = GetInstanceToExecute( _
                            cParentNode, cSubStepRec, cSubStepDtls)
                    If Not cSubStepInst Is Nothing Then
                        Exit For
                    Else
                        ' Continue w/ the next sub-step
                    End If
                Else
                    ' The sub-step is a manager step
                    ' Check if there are any pending instances for
                    ' the manager
                    Set cSubStepInst = mcInstances.QueryPendingInstance( _
                            cParentNode.InstanceId, cSubStepRec.StepId)
                    If cSubStepInst Is Nothing Then
                        ' Find/create an instance to execute
                        Set cSubStepInst = GetInstanceToExecute( _
                                cParentNode, cSubStepRec, cSubStepDtls)
                        If Not cSubStepInst Is Nothing Then
                            Exit For
                        Else
                            ' Continue w/ the next sub-step
                        End If
                    Else
                        ' We have found a pending instance for the
                        ' sub-step (manager) - exit the loop
                        Exit For
                    End If
                End If
            End If
        Next lngIndex

        If lngIndex > cParentNode.SubSteps.Count - 1 Or
cParentNode.SubSteps.Count = 0 Then
            ' If we could not find any sub-steps to execute,
            ' mark the parent node as complete/all started
            Call NoSubStepsToExecute(cParentNode, dtmCompleteTime)
            Set cSubStepInst = Nothing
        End If
    End If

    Set GetSubStepToExecute = cSubStepInst
    Exit Function

GetSubStepToExecuteErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "GetSubStepToExecute"
    Err.Raise vbObjectError + errNavInstancesFailed, mstrSource, _
        LoadResString(errNavInstancesFailed)

End Function

Private Sub TimeCompleteUpdateForStep(cMgrInstance As cInstance, ByVal
EndTime As Currency)

    ' Called when there are no more sub-steps to execute for
    ' the manager step. It updates the end time and status on
    ' the manager.
    Dim lElapsed As Long

    On Error GoTo TimeCompleteUpdateForStepErr

    If cMgrInstance.Key <> mstrDummyRootKey Then
        cMgrInstance.EndTime = EndTime
        cMgrInstance.Status = gintComplete
        lElapsed = (EndTime - cMgrInstance.StartTime) * 10000
        cMgrInstance.ElapsedTime = lElapsed
        RaiseEvent StepComplete(cMgrInstance.Step, EndTime,
cMgrInstance.InstanceId, lElapsed)
    End If

    Exit Sub

TimeCompleteUpdateForStepErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    WriteError errUpdateDisplayFailed, mstrModuleName &
"TimeCompleteUpdateForStep"

End Sub

Private Function GetFreeObject() As Long

    ' Check the array of free objects and retrieve the first one
    If mcFreeSteps.Count > 0 Then
        GetFreeObject = mcFreeSteps(mcFreeSteps.Count - 1)
    Else
        mstrSource = mstrModuleName & "GetFreeObject"
        ShowError errMaxProcessesExceeded
        On Error GoTo 0
        Err.Raise vbObjectError + errMaxProcessesExceeded, _
            mstrSource, _
            LoadResString(errMaxProcessesExceeded)
    End If

End Function
Private Function StepTerminated(cCompleteStep As cStep, ByVal
dtmCompleteTime As Currency, _
        ByVal lngIndex As Long, ByVal InstanceId As Long, ByVal
ExecutionStatus As InstanceStatus) As cStep
    ' This procedure is called whenever a step terminates.
    Dim cTermRec As cTermStep
    Dim cInstRec As cInstance
    Dim cStartInst As cInstance
    Dim lElapsed As Long
    Dim sLogLabel As String
    Dim LogLabels As New cVectorStr
    Dim iItIndex As Long
```

```
   On Error GoTo StepTerminatedErr

   Set cInstRec = mcInstances.QueryInstance(InstanceId)
   If dtmCompleteTime <> 0 And cInstRec.StartTime <> 0 Then
      ' Convert to milliseconds since that is the default precision
      lElapsed = (dtmCompleteTime - cInstRec.StartTime) * 10000
   Else
      lElapsed = 0
   End If

   Set cStartInst = cInstRec
   iItIndex = 0
   Do While cInstRec.Key <> mstrDummyRootKey
      sLogLabel = gstrSQ & cInstRec.Step.StepLabel & gstrSQ

      If iItIndex < cInstRec.Iterators.Count Then
         If cStartInst.Iterators(iItIndex).StepId = cInstRec.Step.StepId Then
            sLogLabel = sLogLabel & msIt & gstrSQ &
cStartInst.Iterators(iItIndex).IteratorName & gstrSQ & _
                  msItValue & gstrSQ & cStartInst.Iterators(iItIndex).Value &
gstrSQ
            iItIndex = iItIndex + 1
         End If
      End If

      If cInstRec.Key = cStartInst.Key Then
         ' Append the execution status
         sLogLabel = sLogLabel & " Status: " & gstrSQ &
gsExecutionStatus(ExecutionStatus) & gstrSQ
         If ExecutionStatus = gintFailed Then
            ' Append the continuation criteria for the step since it failed
            sLogLabel = sLogLabel & " Continuation Criteria: " & gstrSQ &
gsContCriteria(cInstRec.Step.ContinuationCriteria) & gstrSQ
         End If
      End If
      LogLabels.Add sLogLabel

      Set cInstRec = mcInstances.QueryInstance(cInstRec.ParentInstanceId)
   Loop

   Call WriteToWspLog(mintStepComplete, LogLabels, dtmCompleteTime)
   Set LogLabels = Nothing

   ' Adds the terminated step details to a queue.
   Set cTermRec = New cTermStep
   cTermRec.ExecutionStatus = ExecutionStatus
   cTermRec.Index = lngIndex
   cTermRec.InstanceId = InstanceId
   cTermRec.TimeComplete = dtmCompleteTime
   Call mcTermSteps.Add(cTermRec)
   Set cTermRec = Nothing

   RaiseEvent StepComplete(cCompleteStep, dtmCompleteTime, InstanceId,
lElapsed)

   Exit Function

StepTerminatedErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   WriteError errExecuteBranchFailed, mstrSource
   Call ResetForm(lngIndex)

End Function
Public Property Let RootKey(ByVal vdata As String)

   mstrRootKey = vdata

End Property

Public Property Get RootKey() As String
   RootKey = mstrRootKey
End Property

Private Function InitExecStep() As cRunStep
   ' Since arrays of objects cannot be declared as WithEvents,
   ' we use a limited number of objects and set a maximum
   ' on the number of steps that can run in parallel
   ' This is a wrapper that will create an instance of
   ' a cExecuteSM object depending on the index
   Dim lngIndex As Long

   On Error GoTo InitExecStepErr

   lngIndex = GetFreeObject

   Select Case lngIndex + 1
      Case 1
         Set cExecStep1 = New cRunStep
         Set InitExecStep = cExecStep1
      Case 2
         Set cExecStep2 = New cRunStep
         Set InitExecStep = cExecStep2
      Case 3
         Set cExecStep3 = New cRunStep
         Set InitExecStep = cExecStep3
      Case 4
         Set cExecStep4 = New cRunStep
         Set InitExecStep = cExecStep4
      Case 5
         Set cExecStep5 = New cRunStep
         Set InitExecStep = cExecStep5
      Case 6
         Set cExecStep6 = New cRunStep
         Set InitExecStep = cExecStep6
      Case 7
         Set cExecStep7 = New cRunStep
         Set InitExecStep = cExecStep7
      Case 8
         Set cExecStep8 = New cRunStep
         Set InitExecStep = cExecStep8
      Case 9
         Set cExecStep9 = New cRunStep
         Set InitExecStep = cExecStep9
      Case 10
         Set cExecStep10 = New cRunStep
         Set InitExecStep = cExecStep10
      Case 11
         Set cExecStep11 = New cRunStep
         Set InitExecStep = cExecStep11
      Case 12
         Set cExecStep12 = New cRunStep
         Set InitExecStep = cExecStep12
      Case 13
         Set cExecStep13 = New cRunStep
         Set InitExecStep = cExecStep13
      Case 14
         Set cExecStep14 = New cRunStep
         Set InitExecStep = cExecStep14
      Case 15
         Set cExecStep15 = New cRunStep
         Set InitExecStep = cExecStep15
      Case 16
         Set cExecStep16 = New cRunStep
         Set InitExecStep = cExecStep16
      Case 17
         Set cExecStep17 = New cRunStep
         Set InitExecStep = cExecStep17
      Case 18
         Set cExecStep18 = New cRunStep
         Set InitExecStep = cExecStep18
      Case 19
         Set cExecStep19 = New cRunStep
         Set InitExecStep = cExecStep19
      Case 20
         Set cExecStep20 = New cRunStep
         Set InitExecStep = cExecStep20
      Case 21
```

```
    Set cExecStep21 = New cRunStep
    Set InitExecStep = cExecStep21
Case 22
    Set cExecStep22 = New cRunStep
    Set InitExecStep = cExecStep22
Case 23
    Set cExecStep23 = New cRunStep
    Set InitExecStep = cExecStep23
Case 24
    Set cExecStep24 = New cRunStep
    Set InitExecStep = cExecStep24
Case 25
    Set cExecStep25 = New cRunStep
    Set InitExecStep = cExecStep25
Case 26
    Set cExecStep26 = New cRunStep
    Set InitExecStep = cExecStep26
Case 27
    Set cExecStep27 = New cRunStep
    Set InitExecStep = cExecStep27
Case 28
    Set cExecStep28 = New cRunStep
    Set InitExecStep = cExecStep28
Case 29
    Set cExecStep29 = New cRunStep
    Set InitExecStep = cExecStep29
Case 30
    Set cExecStep30 = New cRunStep
    Set InitExecStep = cExecStep30
Case 31
    Set cExecStep31 = New cRunStep
    Set InitExecStep = cExecStep31
Case 32
    Set cExecStep32 = New cRunStep
    Set InitExecStep = cExecStep32
Case 33
    Set cExecStep33 = New cRunStep
    Set InitExecStep = cExecStep33
Case 34
    Set cExecStep34 = New cRunStep
    Set InitExecStep = cExecStep34
Case 35
    Set cExecStep35 = New cRunStep
    Set InitExecStep = cExecStep35
Case 36
    Set cExecStep36 = New cRunStep
    Set InitExecStep = cExecStep36
Case 37
    Set cExecStep37 = New cRunStep
    Set InitExecStep = cExecStep37
Case 38
    Set cExecStep38 = New cRunStep
    Set InitExecStep = cExecStep38
Case 39
    Set cExecStep39 = New cRunStep
    Set InitExecStep = cExecStep39
Case 40
    Set cExecStep40 = New cRunStep
    Set InitExecStep = cExecStep40
Case 41
    Set cExecStep41 = New cRunStep
    Set InitExecStep = cExecStep41
Case 42
    Set cExecStep42 = New cRunStep
    Set InitExecStep = cExecStep42
Case 43
    Set cExecStep43 = New cRunStep
    Set InitExecStep = cExecStep43
Case 44
    Set cExecStep44 = New cRunStep
    Set InitExecStep = cExecStep44
Case 45
    Set cExecStep45 = New cRunStep
    Set InitExecStep = cExecStep45
Case 46
    Set cExecStep46 = New cRunStep
    Set InitExecStep = cExecStep46
Case 47
    Set cExecStep47 = New cRunStep
    Set InitExecStep = cExecStep47
Case 48
    Set cExecStep48 = New cRunStep
    Set InitExecStep = cExecStep48
Case 49
    Set cExecStep49 = New cRunStep
    Set InitExecStep = cExecStep49
Case 50
    Set cExecStep50 = New cRunStep
    Set InitExecStep = cExecStep50
Case 51
    Set cExecStep51 = New cRunStep
    Set InitExecStep = cExecStep51
Case 52
    Set cExecStep52 = New cRunStep
    Set InitExecStep = cExecStep52
Case 53
    Set cExecStep53 = New cRunStep
    Set InitExecStep = cExecStep53
Case 54
    Set cExecStep54 = New cRunStep
    Set InitExecStep = cExecStep54
Case 55
    Set cExecStep55 = New cRunStep
    Set InitExecStep = cExecStep55
Case 56
    Set cExecStep56 = New cRunStep
    Set InitExecStep = cExecStep56
Case 57
    Set cExecStep57 = New cRunStep
    Set InitExecStep = cExecStep57
Case 58
    Set cExecStep58 = New cRunStep
    Set InitExecStep = cExecStep58
Case 59
    Set cExecStep59 = New cRunStep
    Set InitExecStep = cExecStep59
Case 60
    Set cExecStep60 = New cRunStep
    Set InitExecStep = cExecStep60
Case 61
    Set cExecStep61 = New cRunStep
    Set InitExecStep = cExecStep61
Case 62
    Set cExecStep62 = New cRunStep
    Set InitExecStep = cExecStep62
Case 63
    Set cExecStep63 = New cRunStep
    Set InitExecStep = cExecStep63
Case 64
    Set cExecStep64 = New cRunStep
    Set InitExecStep = cExecStep64
Case 65
    Set cExecStep65 = New cRunStep
    Set InitExecStep = cExecStep65
Case 66
    Set cExecStep66 = New cRunStep
    Set InitExecStep = cExecStep66
Case 67
    Set cExecStep67 = New cRunStep
    Set InitExecStep = cExecStep67
Case 68
    Set cExecStep68 = New cRunStep
    Set InitExecStep = cExecStep68
Case 69
    Set cExecStep69 = New cRunStep
    Set InitExecStep = cExecStep69
Case 70
    Set cExecStep70 = New cRunStep
```

```
    Set InitExecStep = cExecStep70
Case 71
    Set cExecStep71 = New cRunStep
    Set InitExecStep = cExecStep71
Case 72
    Set cExecStep72 = New cRunStep
    Set InitExecStep = cExecStep72
Case 73
    Set cExecStep73 = New cRunStep
    Set InitExecStep = cExecStep73
Case 74
    Set cExecStep74 = New cRunStep
    Set InitExecStep = cExecStep74
Case 75
    Set cExecStep75 = New cRunStep
    Set InitExecStep = cExecStep75
Case 76
    Set cExecStep76 = New cRunStep
    Set InitExecStep = cExecStep76
Case 77
    Set cExecStep77 = New cRunStep
    Set InitExecStep = cExecStep77
Case 78
    Set cExecStep78 = New cRunStep
    Set InitExecStep = cExecStep78
Case 79
    Set cExecStep79 = New cRunStep
    Set InitExecStep = cExecStep79
Case 80
    Set cExecStep80 = New cRunStep
    Set InitExecStep = cExecStep80
Case 81
    Set cExecStep81 = New cRunStep
    Set InitExecStep = cExecStep81
Case 82
    Set cExecStep82 = New cRunStep
    Set InitExecStep = cExecStep82
Case 83
    Set cExecStep83 = New cRunStep
    Set InitExecStep = cExecStep83
Case 84
    Set cExecStep84 = New cRunStep
    Set InitExecStep = cExecStep84
Case 85
    Set cExecStep85 = New cRunStep
    Set InitExecStep = cExecStep85
Case 86
    Set cExecStep86 = New cRunStep
    Set InitExecStep = cExecStep86
Case 87
    Set cExecStep87 = New cRunStep
    Set InitExecStep = cExecStep87
Case 88
    Set cExecStep88 = New cRunStep
    Set InitExecStep = cExecStep88
Case 89
    Set cExecStep89 = New cRunStep
    Set InitExecStep = cExecStep89
Case 90
    Set cExecStep90 = New cRunStep
    Set InitExecStep = cExecStep90
Case 91
    Set cExecStep91 = New cRunStep
    Set InitExecStep = cExecStep91
Case 92
    Set cExecStep92 = New cRunStep
    Set InitExecStep = cExecStep92
Case 93
    Set cExecStep93 = New cRunStep
    Set InitExecStep = cExecStep93
Case 94
    Set cExecStep94 = New cRunStep
    Set InitExecStep = cExecStep94
Case 95
    Set cExecStep95 = New cRunStep
    Set InitExecStep = cExecStep95
  Case 96
    Set cExecStep96 = New cRunStep
    Set InitExecStep = cExecStep96
  Case 97
    Set cExecStep97 = New cRunStep
    Set InitExecStep = cExecStep97
  Case 98
    Set cExecStep98 = New cRunStep
    Set InitExecStep = cExecStep98
  Case 99
    Set cExecStep99 = New cRunStep
    Set InitExecStep = cExecStep99
  Case Else
    Set InitExecStep = Nothing
End Select

BugMessage "Sending cExecStep" & (lngIndex + 1) & "!"

If Not InitExecStep Is Nothing Then
  InitExecStep.Index = lngIndex

  ' Remove this element from the collection of free objects
  Call RemoveFreeProcess(lngIndex)
End If

Exit Function

InitExecStepErr:
  ' Log the error code raised by Visual Basic
  Call LogErrors(Errors)
  Set InitExecStep = Nothing

End Function
Public Sub Run()
  ' Calls procedures to build a list of all the steps that
  ' need to be executed and to execute them
  ' Determines whether the run has started/terminated and
  ' raises the Run Start and Complete events.
  Dim cTempStep As cStep

  On Error GoTo RunErr

  If StringEmpty(mstrRootKey) Then
    Call ShowError(errExecuteBranchFailed)
    On Error GoTo 0
    Err.Raise vbObjectError + errExecuteBranchFailed, mstrModuleName &
"Run", _
        LoadResString(errExecuteBranchFailed)
  Else
    ' Execute the first branch
    WriteToWspLog (mintRunStart)
    RaiseEvent RunStart(Determine64BitTime(), mcWspLog.FileName)

    If mcNavSteps.HasChild(StepKey:=mstrRootKey) Then
      Set cTempStep = mcNavSteps.ChildStep(StepKey:=mstrRootKey)
      mstrCurBranchRoot = MakeKeyValid(cTempStep.StepId,
cTempStep.StepType)

      Call CreateDummyInstance(mstrCurBranchRoot)

      ' Run all pending steps in the branch
      If Not RunBranch(mstrCurBranchRoot) Then
        ' Execute a new branch if there aren't any
        ' steps to run
        Call RunNewBranch
      End If
    Else
      WriteToWspLog (mintRunComplete)
      ' No children to execute - the run is complete
      RaiseEvent RunComplete(Determine64BitTime())
    End If
  End If
```

```
    Exit Sub

RunErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Call ShowError(errExecuteBranchFailed, OptArgs:=mstrCurBranchRoot)
    Call ResetForm

End Sub
Private Sub RunNewBranch()
    ' We will build a tree of all instances that occur and
    ' the count of the sub-steps that are running will be
    ' stored at each node in the tree (maintained internally
    ' as an array). Since there can be multiple iterations
    ' of the top level nodes running at the same time, we
    ' create a dummy node at the root that keeps a record of
    ' the instances of the top level node.

    ' Determines whether the run has started/terminated and
    ' raises the Run Start and Complete events.
    Dim cNextStep As cStep
    Dim bRunComplete As Boolean

    On Error GoTo RunNewBranchErr

    bRunComplete = False

    Do
        If StringEmpty(mstrCurBranchRoot) Then
            Exit Do
'           On Error GoTo 0
'           Err.Raise vbObjectError + errExecuteBranchFailed, mstrSource, _
'               LoadResString(errExecuteBranchFailed)
        Else
            Set cNextStep = mcNavSteps.NextStep(StepKey:=mstrCurBranchRoot)
            If cNextStep Is Nothing Then
                mstrCurBranchRoot = gstrEmptyString
                bRunComplete = True
                Exit Do
            Else
                ' Starting execution of a new branch - initialize the
                ' module-level variable
                mstrCurBranchRoot = MakeKeyValid(cNextStep.StepId,
cNextStep.StepType)
                Call CreateDummyInstance(mstrCurBranchRoot)
            End If
        End If
        Debug.Print "Running new branch: " & mstrCurBranchRoot

    ' Loop until we find a branch that has steps to execute
    Loop While Not RunBranch(mstrCurBranchRoot)

    If bRunComplete Then
        WriteToWspLog (mintRunComplete)
        ' Run is complete
        RaiseEvent RunComplete(Determine64BitTime())
    End If

    Exit Sub

RunNewBranchErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Call ShowError(errExecuteBranchFailed, OptArgs:=mstrCurBranchRoot)
    On Error GoTo 0
    mstrSource = mstrModuleName & "RunNewBranch"
    Err.Raise vbObjectError + errExecuteBranchFailed, mstrSource, _
        LoadResString(errExecuteBranchFailed)

End Sub
Private Function RunBranch(strRootNode As String) As Boolean
    ' This procedure is called to run all the necessary steps
    ' in a branch. It can also be called when a step terminates,
    ' in which case the terminated step is passed in as the
    ' optional parameter. When a step terminates, we need to
    ' either wait for some other steps to terminate before
    ' we execute more steps or run as many steps as necessary
    ' Returns True if there are steps currently executing
    ' in the branch, else returns False
    Dim cRunning As cInstance

    On Error GoTo RunBranchErr

    If Not StringEmpty(strRootNode) Then
        ' Call a procedure to execute all the enabled steps
        ' in the branch - will return the step node that is
        ' being executed - nothing means 'No more steps to
        ' execute in the branch'.
        Do
            Set cRunning = RunPendingStepInBranch(strRootNode, cRunning)

        Loop While Not cRunning Is Nothing

        RunBranch = mcDummyRootInstance.IsRunning
    End If

    Exit Function

RunBranchErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "RunBranch"
    Err.Raise vbObjectError + errExecuteBranchFailed, _
        mstrSource, LoadResString(errExecuteBranchFailed)

End Function
Private Sub TimeUpdateForProcess(StepRecord As cStep, _
        ByVal InstanceId As Long, _
        Optional ByVal StartTime As Currency = 0, _
        Optional ByVal EndTime As Currency = 0, _
        Optional ByVal ElapsedTime As Long = 0, _
        Optional Command As String)
    ' We do not maintain start and end timestamps for the constraint
    ' of a step. Hence we check if the process that just started/
    ' terminated is the worker step that is being executed. If so,
    ' we update the start/end time and status on the instance record.

    Dim cInstanceRec As cInstance
    Dim sItVal As String

    On Error GoTo TimeUpdateForProcessErr

    Set cInstanceRec = mcInstances.QueryInstance(InstanceId)

    If StartTime = 0 Then
        RaiseEvent ProcessComplete(StepRecord, EndTime, InstanceId,
ElapsedTime)
    Else
        sItVal = GetInstanceItValue(cInstanceRec)
        RaiseEvent ProcessStart(StepRecord, Command, StartTime, InstanceId, _
            cInstanceRec.ParentInstanceId, sItVal)
    End If

    Call cInstanceRec.UpdateStartTime(StepRecord.StepId, StartTime, EndTime,
ElapsedTime)

    Exit Sub

TimeUpdateForProcessErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    WriteError errUpdateDisplayFailed, mstrModuleName &
"TimeUpdateForProcess"

End Sub
Private Sub TimeStartUpdateForStep(StepRecord As cStep, _
```

```
      ByVal InstanceId As Long, _
      ByVal StartTime As Currency)

   ' Called when a step starts execution. Checks if this is the
   ' first enabled child of the manager step. If so, updates
   ' the start time and status on the manager.
   ' Also raises the Step Start event for the completed step.

   Dim cStartInst As cInstance
   Dim cInstanceRec As cInstance
   Dim LogLabels As New cVectorStr
   Dim iItIndex As Long
   Dim sLogLabel As String
   Dim sPath As String
   Dim sIt As String
   Dim sItVal As String

   On Error GoTo TimeStartUpdateForStepErr

   Set cStartInst = mcInstances.QueryInstance(InstanceId)

   ' Determine the step path and iterator values for the step and raise a step start
event
   Set cInstanceRec = cStartInst
   Do While cInstanceRec.Key <> mstrDummyRootKey
      If Not StringEmpty(sPath) Then
         sPath = sPath & gstrFileSeparator
      End If
      sPath = sPath & gstrSQ & cInstanceRec.Step.StepLabel & gstrSQ
      Set cInstanceRec =
mcInstances.QueryInstance(cInstanceRec.ParentInstanceId)
   Loop

   For iItIndex = cStartInst.Iterators.Count - 1 To 0 Step -1
      If Not StringEmpty(sIt) Then
         sIt = sIt & gstrFileSeparator
      End If
      sIt = sIt & gstrSQ & cStartInst.Iterators(iItIndex).Value & gstrSQ
   Next iItIndex

   sItVal = GetInstanceItValue(cStartInst)
   RaiseEvent StepStart(StepRecord, StartTime, InstanceId,
cStartInst.ParentInstanceId, _
         sPath, sIt, sItVal)

   iItIndex = 0
   Set cInstanceRec = cStartInst
   ' Raise a StepStart event for the manager step, if this is it's first sub-step being
executed
   Do While cInstanceRec.Key <> mstrDummyRootKey

      sLogLabel = gstrSQ & cInstanceRec.Step.StepLabel & gstrSQ
      If iItIndex < cStartInst.Iterators.Count Then
         If cStartInst.Iterators(iItIndex).StepId = cInstanceRec.Step.StepId Then
            sLogLabel = sLogLabel & msIt & gstrSQ &
cStartInst.Iterators(iItIndex).IteratorName & gstrSQ & _
                  msItValue & gstrSQ & cStartInst.Iterators(iItIndex).Value &
gstrSQ
            iItIndex = iItIndex + 1
         End If
      End If
      LogLabels.Add sLogLabel

      If cInstanceRec.Key <> cStartInst.Key And cInstanceRec.StartTime = 0
Then
         cInstanceRec.StartTime = StartTime
         cInstanceRec.Status = gintRunning
         sItVal = GetInstanceItValue(cInstanceRec)
         ' The step path and iterator values are not needed for manager steps, since
         ' they are primarily used by the run status form
         RaiseEvent StepStart(cInstanceRec.Step, StartTime,
cInstanceRec.InstanceId, _
               cInstanceRec.ParentInstanceId, gstrEmptyString, gstrEmptyString, _
               sItVal)
      End If

      Set cInstanceRec =
mcInstances.QueryInstance(cInstanceRec.ParentInstanceId)
   Loop

   Call WriteToWspLog(mintStepStart, LogLabels, StartTime)
   Set LogLabels = Nothing

   Exit Sub

TimeStartUpdateForStepErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   WriteError errUpdateDisplayFailed, mstrModuleName &
"TimeStartUpdateForStep"

End Sub
Private Sub WriteToWspLog(iLogEvent As WspLogEvents, Optional StepDtls
As cVectorStr, _
      Optional dtStamp As Currency = gdtmEmpty)

   ' Writes to the workspace log that is generated for the run. The last three
   ' parameters are valid only for Step Start and Step Complete events.
   Static bError As Boolean
   Dim sLabel As String
   Dim lIndex As Long
   Dim bHdr As Boolean
   Dim cTempConn As cConnection

   On Error GoTo WriteToWspLogErr

   Select Case iLogEvent
      Case mintRunStart
         Set mcWspLog = New cFileSM
         mcWspLog.FileName = GetDefaultDir(WspId, mcParameters) &
gstrFileSeparator & _
               Trim(Str(RunId)) & gstrFileSeparator & "SMLog-" & Format(Now,
FMT_WSP_LOG_FILE) & gstrLogFileSuffix
         mcWspLog.WriteLine (JulianDateToString(Determine64BitTime()) & "
Start Run: " & vbTab & gstrSQ & GetWorkspaceDetails(WorkspaceId:=WspId))
& gstrSQ

         ' Write all current parameter values to the log
         bHdr = False
         For lIndex = 0 To mcParameters.ParameterCount - 1
            If mcParameters(lIndex).ParameterType <> gintParameterApplication
Then
               If Not bHdr Then
                  mcWspLog.WriteField
JulianDateToString(Determine64BitTime()) & " Parameters: "
                  bHdr = True
               Else
                  mcWspLog.WriteField vbTab & vbTab & vbTab
               End If
               mcWspLog.WriteLine vbTab & gstrSQ &
mcParameters(lIndex).ParameterName & gstrSQ & vbTab & vbTab & gstrSQ &
mcParameters(lIndex).ParameterValue & gstrSQ
            End If
         Next lIndex

         ' Write all connection properties to the log
         For lIndex = 0 To RunConnections.Count - 1
            Set cTempConn = RunConnections(lIndex)
            If lIndex = 0 Then
               mcWspLog.WriteField JulianDateToString(Determine64BitTime())
& " Connections: "
            Else
               mcWspLog.WriteField vbTab & vbTab & vbTab
            End If
            mcWspLog.WriteLine vbTab & gstrSQ &
cTempConn.ConnectionName & gstrSQ & _
                  vbTab & vbTab & gstrSQ & cTempConn.ConnectionValue &
gstrSQ & _
```

```
                vbTab & "No Count: " & gstrSQ & cTempConn.NoCountDisplay
& gstrSQ & gstrBlank & _
                "No Execute: " & gstrSQ & cTempConn.NoExecute & gstrSQ &
gstrBlank & _
                "Parse Query Only: " & gstrSQ & cTempConn.ParseQueryOnly &
gstrSQ & gstrBlank & _
                "Quoted Identifiers: " & gstrSQ & cTempConn.QuotedIdentifiers
& gstrSQ & gstrBlank & _
                "ANSI Nulls: " & gstrSQ & cTempConn.AnsiNulls & gstrSQ &
gstrBlank & _
                "Show Query Plan: " & gstrSQ & cTempConn.ShowQueryPlan &
gstrSQ & gstrBlank & _
                "Show Stats Time: " & gstrSQ & cTempConn.ShowStatsTime &
gstrSQ & gstrBlank & _
                "Show Stats IO: " & gstrSQ & cTempConn.ShowStatsIO & gstrSQ
& gstrBlank & _
                "Row Count" & gstrSQ & cTempConn.RowCount & gstrSQ &
gstrBlank & _
                "Query Timeout" & gstrSQ & cTempConn.QueryTimeOut &
gstrSQ
        Next lIndex

      Case mintRunComplete
        BugAssert Not mcWspLog Is Nothing
        mcWspLog.WriteLine (JulianDateToString(Determine64BitTime()) & "
Comp. Run: " & vbTab & gstrSQ &
GetWorkspaceDetails(WorkspaceId:=WspId)) & gstrSQ
        Set mcWspLog = Nothing

      Case mintStepStart
        For lIndex = StepDtls.Count - 1 To 0 Step -1
          sLabel = StepDtls(lIndex)
          If lIndex = StepDtls.Count - 1 Then
            mcWspLog.WriteLine JulianDateToString(dtStamp) & " Start Step: "
& vbTab & sLabel
          Else
            mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab & sLabel
          End If
        Next lIndex

      Case mintStepComplete
        For lIndex = StepDtls.Count - 1 To 0 Step -1
          sLabel = StepDtls(lIndex)
          If lIndex = StepDtls.Count - 1 Then
            mcWspLog.WriteLine JulianDateToString(dtStamp) & " Comp. Step:
" & vbTab & sLabel
          Else
            mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab & sLabel
          End If
        Next lIndex

  End Select

  Exit Sub

WriteToWspLogErr:
  If Not bError Then
    bError = True
  End If

End Sub
'Private Sub WriteToWspLog(iLogEvent As WspLogEvents, Optional StepDtls
As cVectorStr, _
'      Optional dtStamp As Date = gdtmEmpty)
'
'  This function uses the LogWriter dll - memory corruption problems since the
vb exe
'  and the vc Execute Dll both use the same dll to write.
'   ' Writes to the workspace log that is generated for the run. The last three
'   ' parameters are valid only for StepStart and StepComplete events.
'   Static bError As Boolean
'   Static sFile As String
'   Dim sLabel As String
'   Dim lIndex As Long
'   Dim bHdr As Boolean
'
'   On Error GoTo WriteToWspLogErr
'
'   Select Case iLogEvent
'      Case mintRunStart
'        Set mcWspLog = New LOGWRITERLib.SMLog
'        sFile = App.Path & "\" & "SMLog-" & Format(Now,
FMT_WSP_LOG_FILE) & gstrLogFileSuffix
'        mcWspLog.FileName = sFile
'        mcWspLog.Init
'        mcWspLog.WriteLine (Format(Now, FMT_WSP_LOG_DATE) & " Start
Run: " & vbTab & gstrSQ & GetWorkspaceDetails(WorkspaceId:=WspId)) &
gstrSQ
'
'        ' Write all current parameter values to the log
'        bHdr = False
'        For lIndex = 0 To mcParameters.ParameterCount - 1
'          If mcParameters(lIndex).ParameterType <> gintParameterApplication
Then
'            If Not bHdr Then
'              'mcWspLog.WriteLine Format(Now, FMT_WSP_LOG_DATE) &
" Parameters: " & vbTab & gstrSQ & mcParameters(lIndex).ParameterName &
gstrSQ & vbTab & vbTab & gstrSQ & mcParameters(lIndex).ParameterValue &
gstrSQ
'              bHdr = True
'            Else
'              'mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab &
gstrSQ & mcParameters(lIndex).ParameterName & gstrSQ & vbTab & vbTab &
gstrSQ & mcParameters(lIndex).ParameterValue & gstrSQ
'            End If
'          End If
'        Next lIndex
'
'      Case mintRunComplete
'        BugAssert Not mcWspLog Is Nothing
'        mcWspLog.WriteLine (Format(Now, FMT_WSP_LOG_DATE) & "
Comp. Run: " & vbTab & gstrSQ &
GetWorkspaceDetails(WorkspaceId:=WspId)) & gstrSQ
'        Set mcWspLog = Nothing
'
'      Case mintStepStart
'        For lIndex = StepDtls.Count - 1 To 0 Step -1
'          sLabel = StepDtls(lIndex)
'          If lIndex = StepDtls.Count - 1 Then
'            mcWspLog.WriteLine Format(dtStamp, FMT_WSP_LOG_DATE)
& " Start Step: " & vbTab & sLabel
'          Else
'            mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab & sLabel
'          End If
'        Next lIndex
'
'      Case mintStepComplete
'        For lIndex = StepDtls.Count - 1 To 0 Step -1
'          sLabel = StepDtls(lIndex)
'          If lIndex = StepDtls.Count - 1 Then
'            mcWspLog.WriteLine Format(dtStamp, FMT_WSP_LOG_DATE)
& " Comp. Step: " & vbTab & sLabel
'          Else
'            mcWspLog.WriteLine vbTab & vbTab & vbTab & vbTab & sLabel
'          End If
'        Next lIndex
'
'   End Select
'
'   Exit Sub
'
'WriteToWspLogErr:
'   If Not bError Then
'      bError = True
'   End If
'
'End Sub
'
```

```
Public Property Get WspPreExecution() As Variant
   WspPreExecution = mcvntWspPreCons
End Property
Public Property Let WspPreExecution(ByVal vdata As Variant)
   mcvntWspPreCons = vdata
End Property

Public Property Get WspPostExecution() As Variant
   WspPostExecution = mcvntWspPostCons
End Property
Public Property Let WspPostExecution(ByVal vdata As Variant)
   mcvntWspPostCons = vdata
End Property

Private Sub ExecuteStep(cCurStep As cInstance)
   ' Initializes a cRunStep object with all the properties
   ' corresponding to the step to be executed and calls it's
   ' execute method to execute the step

   Dim cExecStep As cRunStep

   On Error GoTo ExecuteStepErr
   mstrSource = mstrModuleName & "ExecuteStep"

   ' Confirm that the step is a worker
   If cCurStep.Step.StepType <> gintWorkerStep Then
      On Error GoTo 0
      Err.Raise vbObjectError + errExecInstanceFailed, mstrSource, _
            LoadResString(errExecInstanceFailed)
   End If

   Set cExecStep = InitExecStep()
   ' Exceeded the number of processes that we can run simultaneously
   If cExecStep Is Nothing Then
      ' Raise an error
      On Error GoTo 0
      Err.Raise vbObjectError + errProgramError, mstrSource, _
         LoadResString(errProgramError)
   End If
   ' Initialize the instance id - not needed for step execution
   ' but necessary to identify later which instance completed
   cExecStep.InstanceId = cCurStep.InstanceId

   Set cExecStep.ExecuteStep = cCurStep.Step
   Set cExecStep.Iterators = cCurStep.Iterators
   Set cExecStep.Globals = mcRunSteps
   Set cExecStep.WspParameters = mcParameters
   Set cExecStep.WspConnections = RunConnections
   Set cExecStep.WspConnDtls = RunConnDtls

   ' Initialize all the pre and post-execution constraints that
   ' have been defined globally for the workspace
   cExecStep.WspPreCons = mcvntWspPreCons
   cExecStep.WspPostCons = mcvntWspPostCons

   ' Initialize all the pre and post-execution constraints for
   ' the step being executed
   cExecStep.PreCons = DetermineConstraints(cCurStep, gintPreStep)
   cExecStep.PostCons = DetermineConstraints(cCurStep, gintPostStep)

   cExecStep.RunId = RunId
   cExecStep.CreateInputFiles = CreateInputFiles

   ' Call the execute method to execute the step
   cExecStep.Execute

   Set cExecStep = Nothing

   Exit Sub

ExecuteStepErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Call ExecutionFailed(cExecStep)

End Sub

Public Property Set Steps(cRunSteps As cArrSteps)

   Set mcRunSteps = cRunSteps
   Set mcNavSteps.StepRecords = cRunSteps

End Property
Public Property Set Parameters(cParameters As cArrParameters)
   ' A reference to the parameter array - we use it to
   ' substitute parameter values in the step text

   Set mcParameters = cParameters

End Property
Public Property Get Steps() As cArrSteps

   Set Steps = mcRunSteps

End Property
Public Property Get Constraints() As cArrConstraints

   Set Constraints = mcRunConstraints

End Property
Public Property Set Constraints(vdata As cArrConstraints)

   Set mcRunConstraints = vdata

End Property


Private Sub cExecStep1_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep1_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep1_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep1.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep1_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep9_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub
```

```
Private Sub cExecStep9_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep9_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep9.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep9_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep10_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep10_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep10_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep10.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep10_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep11_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep11_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep11_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep11.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep11_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep12_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep12_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep12_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep12.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep12_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep13_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep13_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep13_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep13.Index,
InstanceId, Status)

End Sub
```

```
Private Sub cExecStep13_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep14_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep14_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep14_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep14.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep14_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep15_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep15_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep15_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep15.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep15_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep16_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep16_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep16_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep16.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep16_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
Private Sub cExecStep17_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep17_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep17_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep17.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep17_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep18_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep18_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)
```

```
   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep18_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep18.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep18_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep19_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub


Private Sub cExecStep19_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep19_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep19.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep19_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep20_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep20_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep20_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep20.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep20_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep21_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep21_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep21_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep21.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep21_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep22_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep22_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep22_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep22.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep22_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
```

```
Private Sub cExecStep23_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep23_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep23_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep23.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep23_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep24_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep24_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep24_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep24.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep24_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep25_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep25_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep25_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep25.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep25_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep26_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep26_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep26_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep26.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep26_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep27_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep27_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep27_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep27.Index,
InstanceId, Status)

End Sub
```

```
Private Sub cExecStep27_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep28_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep28_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep28_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep28.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep28_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep29_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep29_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep29_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep29.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep29_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep30_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep30_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep30_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep30.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep30_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep31_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep31_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep31_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep31.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep31_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep32_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep32_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub
```

```
Private Sub cExecStep32_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep32.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep32_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep33_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep33_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep33_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep33.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep33_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep34_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep34_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep34_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep34.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep34_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep35_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep35_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep35_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep35.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep35_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep36_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep36_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep36_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep36.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep36_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep37_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub
```

```
Private Sub cExecStep37_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep37_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep37.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep37_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep38_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep38_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep38_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep38.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep38_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep39_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep39_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep39_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep39.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep39_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep40_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep40_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep40_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep40.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep40_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep41_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep41_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep41_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep41.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep41_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
```

```
Private Sub cExecStep42_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep42_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep42_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep42.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep42_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep43_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep43_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep43_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep43.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep43_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep44_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep44_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep44_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep44.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep44_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep45_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep45_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep45_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep45.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep45_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep46_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep46_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep46_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep46.Index,
InstanceId, Status)

End Sub
```

```
Private Sub cExecStep46_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep47_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep47_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep47_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep47.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep47_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep48_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep48_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep48_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep48.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep48_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep49_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep49_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep49_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep49.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep49_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep50_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep50_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep50_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep50.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep50_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep51_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep51_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub
```

```
Private Sub cExecStep51_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep51.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep51_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep52_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep52_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep52_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep52.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep52_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep53_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep53_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep53_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep53.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep53_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep54_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep54_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep54_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep54.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep54_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep55_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep55_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep55_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep55.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep55_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep56_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub
```

```
Private Sub cExecStep56_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep56_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep56.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep56_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep57_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep57_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep57_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep57.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep57_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep58_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep58_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep58_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep58.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep58_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep59_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep59_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep59_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep59.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep59_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep60_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep60_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep60_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep60.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep60_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
```

```
Private Sub cExecStep61_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep61_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep61_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep61.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep61_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep62_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep62_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep62_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep62.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep62_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep63_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep63_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep63_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep63.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep63_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep64_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep64_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep64_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep64.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep64_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep65_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep65_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep65_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep65.Index,
InstanceId, Status)

End Sub
```

```
Private Sub cExecStep65_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep66_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep66_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep66_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep66.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep66_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep67_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep67_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep67_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep67.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep67_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep68_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep68_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep68_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep68.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep68_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep69_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep69_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep69_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep69.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep69_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep70_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep70_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub
```

```
Private Sub cExecStep70_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep70.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep70_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep71_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep71_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep71_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep71.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep71_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep72_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep72_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep72_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep72.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep72_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep73_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep73_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep73_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep73.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep73_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep74_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep74_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep74_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep74.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep74_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep75_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub
```

```
Private Sub cExecStep75_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep75_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep75.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep75_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep76_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep76_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep76_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep76.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep76_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep77_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep77_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep77_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep77.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep77_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep78_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep78_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep78_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep78.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep78_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep79_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep79_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep79_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep79.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep79_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
```

```
Private Sub cExecStep80_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep80_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep80_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep80.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep80_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep81_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep81_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep81_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep81.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep81_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep82_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep82_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep82_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep82.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep82_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep83_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep83_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep83_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep83.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep83_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep84_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep84_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep84_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep84.Index,
InstanceId, Status)

End Sub
```

```
Private Sub cExecStep84_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep85_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep85_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep85_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep85.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep85_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep86_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep86_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep86_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep86.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep86_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep87_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep87_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep87_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep87.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep87_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep88_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep88_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep88_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep88.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep88_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep89_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep89_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub
```

```
Private Sub cExecStep89_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep89.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep89_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep90_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep90_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep90_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep90.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep90_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep91_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep91_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep91_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep91.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep91_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep92_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep92_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep92_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep92.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep92_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep93_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep93_ProcessStart(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep93_StepComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

  Call StepTerminated(cStepRecord, dtmEndTime, cExecStep93.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep93_StepStart(cStepRecord As cStep, _
    dtmStartTime As Currency, InstanceId As Long)

  Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep94_ProcessComplete(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub
```

```
Private Sub cExecStep94_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep94_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep94.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep94_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep95_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep95_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep95_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep95.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep95_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep96_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep96_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep96_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep96.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep96_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep97_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep97_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep97_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep97.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep97_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep98_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep98_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep98_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep98.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep98_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub
```

```
Private Sub cExecStep99_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep99_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep99_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep99.Index,
InstanceId, Status)

End Sub

Private Sub cExecStep99_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep2_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep2_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep2_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep2.Index, _
         InstanceId, Status)

End Sub

Private Sub cExecStep2_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub


Private Sub cExecStep3_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep3_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub


Private Sub cExecStep3_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep3.Index, _
         InstanceId, Status)

End Sub

Private Sub cExecStep3_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub


Private Sub cExecStep4_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep4_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep4_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep4.Index, _
         InstanceId, Status)

End Sub


Private Sub cExecStep4_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep5_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep5_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep5_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)
```

```
   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep5.Index, _
         InstanceId, Status)

End Sub

Private Sub cExecStep5_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep6_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep6_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep6_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep6.Index, _
         InstanceId, Status)

End Sub

Private Sub cExecStep6_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep7_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep7_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep7_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep7.Index, _
         InstanceId, Status)

End Sub

Private Sub cExecStep7_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub cExecStep8_ProcessComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
EndTime:=dtmEndTime, ElapsedTime:=lElapsed)

End Sub

Private Sub cExecStep8_ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, lngInstanceId As Long)

   Call TimeUpdateForProcess(cStepRecord, lngInstanceId,
StartTime:=dtmStartTime, Command:=strCommand)

End Sub

Private Sub cExecStep8_StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)

   Call StepTerminated(cStepRecord, dtmEndTime, cExecStep8.Index, _
         InstanceId, Status)

End Sub

Private Sub cExecStep8_StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)

   Call TimeStartUpdateForStep(cStepRecord, InstanceId, dtmStartTime)

End Sub

Private Sub Class_Initialize()

   Dim lngCount As Long
   Dim lngTemp As Long

   On Error GoTo InitializeErr

   Set mcFreeSteps = New cVectorLng
   ' Initialize the array of free objects with all elements
   ' for now
   For lngCount = 0 To glngNumConcurrentProcesses - 1 Step 1
      mcFreeSteps.Add lngCount
   Next lngCount

   ' Initialize a byte array with the number of free processes. It will
   ' be used later to determine if any step is running
   ' Each element in the array can represent 8 steps, 1 for each bit
   ReDim mbarrFree(glngNumConcurrentProcesses \ gintBitsPerByte)

   ' Initialize each element in the byte array w/ all 1's
   ' (upto glngNumConcurrentProcesses)
   For lngCount = LBound(mbarrFree) To UBound(mbarrFree) Step 1
      lngTemp = IIf( _
            glngNumConcurrentProcesses - (gintBitsPerByte * lngCount) >
gintBitsPerByte, _
            gintBitsPerByte, _
            glngNumConcurrentProcesses - (gintBitsPerByte * lngCount))

      mbarrFree(lngCount) = (2 ^ lngTemp) - 1
   Next lngCount

   Set mcInstances = New cInstances
   Set mcFailures = New cFailedSteps
   Set mcNavSteps = New cStepTree
   Set mcTermSteps = New cTermSteps

   ' Initialize the Abort flag to False
   mblnAbort = False
   mblnAsk = False

   Exit Sub

InitializeErr:
```

```
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errInitializeFailed, mstrModuleName & "Initialize",
_
            LoadResString(errInitializeFailed)

End Sub
Private Sub Class_Terminate()

    On Error GoTo Class_TerminateErr

    mcFreeSteps.Clear
    Set mcFreeSteps = Nothing
    ReDim mbarrFree(0)

    mcInstances.Clear
    Set mcInstances = Nothing

    Set mcFailures = Nothing
    Set mcNavSteps = Nothing
    Set mcTermSteps = Nothing

    Exit Sub

Class_TerminateErr:
    Call LogErrors(Errors)

End Sub

Private Sub mcTermSteps_TermStepExists(cStepDetails As cTermStep)

    Call RunNextStep(cStepDetails.TimeComplete, cStepDetails.Index, _
            cStepDetails.InstanceId, cStepDetails.ExecutionStatus)

End Sub
```

cRunItDetails.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunItDetails"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cRunItDetails.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module encapsulates the properties of iterator values
'              that are used by the step being executed at runtime.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'

Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cRunItDetails."
Private mstrSource As String

Private mstrIteratorName As String
Private mintType As ValueType
Private mlngSequence As Long
Private mlngFrom As Long
Private mlngTo As Long
Private mlngStep As Long
Private mstrValue As String

Public Property Get RangeTo() As Long

    RangeTo = mlngTo

End Property
Public Property Let RangeTo(ByVal vdata As Long)

    mlngTo = vdata

End Property

Public Property Get RangeFrom() As Long

    RangeFrom = mlngFrom

End Property
Public Property Get Sequence() As Long

    Sequence = mlngSequence

End Property

Public Property Get RangeStep() As Long

    RangeStep = mlngStep

End Property
Public Property Let RangeStep(vdata As Long)

    mlngStep = vdata

End Property

Public Property Let RangeFrom(ByVal vdata As Long)

    mlngFrom = vdata

End Property
Public Property Let Sequence(ByVal vdata As Long)

    mlngSequence = vdata

End Property

Public Property Get IteratorType() As ValueType

    IteratorType = mintType

End Property
Public Property Let IteratorType(ByVal vdata As ValueType)

    On Error GoTo TypeErr
    mstrSource = mstrModuleName & "Type"

    ' These constants have been defined in the enumeration,
    ' Type, which is exposed
    Select Case vdata
        Case gintFrom, gintTo, gintStep, gintValue
            mintType = vdata

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errTypeInvalid, _
                mstrSource, LoadResString(errTypeInvalid)
    End Select

    Exit Property

TypeErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Type"
    On Error GoTo 0
    Err.Raise vbObjectError + errTypeInvalid, _
```

```
    mstrSource, LoadResString(errTypeInvalid)

End Property
Private Sub IsList()

  If mintType <> gintValue Then
    On Error GoTo 0
    Err.Raise vbObjectError + errInvalidProperty, mstrSource, _
        LoadResString(errInvalidProperty)
  End If

End Sub
Private Sub IsRange()

  If mintType = gintValue Then
    On Error GoTo 0
    Err.Raise vbObjectError + errInvalidProperty, mstrSource, _
        LoadResString(errInvalidProperty)
  End If

End Sub

Public Property Get Value() As String

  Value = mstrValue

End Property
Public Property Let Value(vdata As String)

  mstrValue = vdata

End Property


Public Property Get IteratorName() As String

  IteratorName = mstrIteratorName

End Property
Public Property Let IteratorName(ByVal vdata As String)

  mstrIteratorName = vdata

End Property
```

cRunItNode.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunItNode"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
' An iterator class containing the properties that are used
' by the stpe being executed.
' These iterators might actually come from steps that are at
' a higher level than the step actually being executed (viz.
' direct ascendants of the step at any level).

Option Explicit

Public IteratorName As String
Public Value As String
Public StepId As Long
```

cRunOnly.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunOnly"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Option Explicit

Public Event Done()
Private WithEvents mcRunWsp As cRunWorkspace
Attribute mcRunWsp.VB_VarHelpID = -1

Public WspName As String
Public WorkspaceId As Long
Public WspLog As String

Public Sub RunWsp()

  On Error GoTo RunWspErr

  Set mcRunWsp = New cRunWorkspace
  Set mcRunWsp.LoadDb = dbsAttTool
  mcRunWsp.WorkspaceId = WorkspaceId
  mcRunWsp.CreateInputFiles = True
  mcRunWsp.RunWorkspace

  Exit Sub

RunWspErr:
  ' Log the VB error code
  LogErrors Errors

End Sub

Private Sub mcRunWsp_RunComplete(dtmEndTime As Currency)

  MsgBox "Completed executing workspace: " & gstrSQ & WspName & gstrSQ
& " at " & _
      JulianDateToString(dtmEndTime) & "." & vbCrLf & vbCrLf & _
      "The log file for the run is: " & gstrSQ & WspLog & gstrSQ & "."
  RaiseEvent Done

End Sub

Private Sub mcRunWsp_RunStart(dtmStartTime As Currency, strWspLog As
String, lRunId As Long)
  WspLog = strWspLog
End Sub
```

cRunStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:      cRunStep.cls
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:   This class executes the step that is assigned to the
'             ExecuteStep property. It executes the pre-execution constraints
'             in sequence and then the step itself. At the end it executes
'             the post-execution constraints. Since these steps should always
'             be executed in sequence, each step is only fired on the
'             completion of the previous step.
'  Contact:   Reshma Tharamal (reshmat@microsoft.com)
```

```
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cRunStep."
Private mstrSource As String

' Local variable(s) to hold property value(s)
Private mcStep As cStep
Private mcGlobals As cArrSteps
Private mcvntWspPreCons As Variant
Private mcvntWspPostCons As Variant
Private mcvntPreCons As Variant
Private mcvntPostCons As Variant
Private mcIterators As cRunColIt
Private mlngInstanceId As Long ' Identifier for the current instance
Private mlngIndex As Long ' Index value for the current instance
Private mstrCommand As String ' The command string
Private msRunStepDtl As String ' Step text/file name that will go into the
run_step_details table
Private mblnAbort As Boolean   ' Set to True when the user aborts the run
Private msOutputFile As String
Private msErrorFile As String
Private miStatus As InstanceStatus
Private mcVBErr As cVBErrorsSM
Public WspParameters As cArrParameters
Public WspConnections As cConnections
Public WspConnDtls As cConnDtls

Private WithEvents mcTermProcess As cTermProcess
Attribute mcTermProcess.VB_VarHelpID = -1
Public RunId As Long
Public CreateInputFiles As Boolean
Private msOutputDir As String

' Object that will execute the step
Private WithEvents mcExecObj As EXECUTEDLLLib.Execute
Attribute mcExecObj.VB_VarHelpID = -1

' Holds the step that is currently being executed (constraint or
' worker step)
Private mcExecStep As cStep

Private Const msCompareExe As String = "\diff.exe"

Private Enum NextNodeType
   mintWspPreConstraint = 1
   mintPreConstraint
   mintStep
   mintWspPostConstraint
   mintPostConstraint
End Enum

' Public events to notify the calling function of the
' start and end time for each step
Public Event StepStart(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long)
Public Event StepComplete(cStepRecord As cStep, _
      dtmEndTime As Currency, InstanceId As Long, Status As InstanceStatus)
Public Event ProcessStart(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, _
      InstanceId As Long)
Public Event ProcessComplete(cStepRecord As cStep, _
      dtmStartTime As Currency, InstanceId As Long, lElapsed As Long)

Private Function AppendDiffErrors(sDiffFile As String)
   ' The file containing the errors generated by the diff utility is passed in
   ' These errors are appended to the error file for the step

   Dim sTemp As String
   Dim InputFile As Integer

   If Not StringEmpty(sDiffFile) Then
      InputFile = FreeFile
      Open sDiffFile For Input Access Read As InputFile

      Do While Not EOF(InputFile)        ' Loop until end of file.
         Line Input #InputFile, sTemp    ' Read line into variable.
         mcVBErr.LogMessage sTemp
      Loop

      Close InputFile
   End If

End Function

Private Sub CreateStepTextFile()
   ' Creates a file containing the step text being executed
   On Error GoTo CreateStepTextFileErr

   Dim sInputFile As String

   If mcExecStep.ExecutionMechanism = gintExecuteShell Then
      sInputFile = GetOutputFile(gsCmdFileSuffix)
   Else
      sInputFile = GetOutputFile(gsSqlFileSuffix)
   End If

   ' Generate a file containing the step text being executed
   If Not StringEmpty(mcExecStep.StepTextFile) Or
mcExecStep.ExecutionMechanism = gintExecuteShell Then
      FileCopy mstrCommand, sInputFile
   Else
      Call WriteCommandToFile(mstrCommand, sInputFile)
   End If

   Exit Sub

CreateStepTextFileErr:

   mcVBErr.LogVBErrors

End Sub
Private Function GetOutputFile(strFileExt As String) As String
   ' This function generates the output file name for the step currently being
executed
   ' The value of the built-in parameter 'DefaultDir' is appended with the run
identifier
   ' for the file location
   ' The step label is used for the file name and a combination of all iterator values
   ' for the step is used to make the output files unique for each instance
   Dim sFile As String
   Dim sIt As String
   Dim lIt As Long

   On Error GoTo GetOutputFileErr

   sFile = SubstituteParametersIfPossible(mcExecStep.StepLabel)

   sFile = TranslateStepLabel(sFile)

   If mcExecStep Is mcStep Then
      ' Use iterators that have been defined for the worker or any of it's managers
      ' to make the error/log file unique for this instance
      For lIt = mcIterators.Count - 1 To 0 Step -1
         sIt = sIt & gsExtSeparator & mcIterators(lIt).Value
      Next lIt
   End If
   sIt = sIt & strFileExt

   ' Ensure that the length of the complete path does not exceed 255 characters
   If Len(msOutputDir) + Len(sFile) + Len(sIt) > MAX_PATH Then
      sFile = Mid(sFile, 1, MAX_PATH - Len(sIt) - Len(msOutputDir))
   End If
   GetOutputFile = msOutputDir & sFile & sIt
   Exit Function
```

```
GetOutputFileErr:

   ' Does not make sense to log error to the error file yet. Write to the project
   ' log and return the step label as default
   GetOutputFile = mcExecStep.StepLabel & gsExtSeparator & strFileExt

End Function


Private Sub HandleExecutionError()

   On Error GoTo HandleExecutionError

   ' Log the error code raised by Visual Basic
   miStatus = gintFailed
   mcVBErr.LogVBErrors
   Call mcVBErr.WriteError(errExecuteStepFailed, _
         OptArgs:="Continuation criteria for the step is: " &
gsContCriteria(mcStep.ContinuationCriteria))

HandleExecutionError:

   ' Logging failed - return

End Sub

Public Property Get Index() As Long

   Index = mlngIndex

End Property
Public Property Let Index(ByVal vdata As Long)

   mlngIndex = vdata

End Property
Private Function InitializeExecStatus() As InstanceStatus
   Dim sCompareFile As String

   On Error GoTo InitializeExecStatusErr

   InitializeExecStatus = mcExecObj.StepStatus

   If InitializeExecStatus = gintComplete Then
      If Not StringEmpty(mcExecStep.FailureDetails) Then
         ' Compare output to determine whether the step failed
         sCompareFile = GetShortName(SubstituteParameters( _
               mcExecStep.FailureDetails, mcExecStep.WorkspaceId, mcIterators, _
               WspParameters))
         InitializeExecStatus = IIf(CompareOutput(sCompareFile, msOutputFile),
gintComplete, gintFailed)
      End If
   End If

   Exit Function

InitializeExecStatusErr:
   mcVBErr.LogVBErrors
'    Call LogErrors(Errors)
   InitializeExecStatus = mcExecObj.StepStatus

End Function
Private Function CompareOutput(sCompareFile As String, sOutputFile As String)
As Boolean

   Dim sCmpOutput As String
   Dim sDiffOutput As String

   On Error GoTo CompareOutputErr

   ' Create temporary files to store the file compare output and
   ' the errors generated by the compare function
   sCmpOutput = CreateTempFile()
   sDiffOutput = CreateTempFile()

   ' Run the compare utility and redirect it's output and errors
   SyncShell ("cmd /c " & _
         GetShortName(App.Path & msCompareExe) & gstrBlank & _
         sCompareFile & gstrBlank & sOutputFile & _
         " > " & sCmpOutput & " 2> " & sDiffOutput)

   If FileLen(sDiffOutput) > 0 Then
      ' The compare generated errors - append error msgs to the error file
      Call AppendDiffErrors(sDiffOutput)
      CompareOutput = False
   Else
      CompareOutput = (FileLen(sCmpOutput) = 0)
   End If

   If Not CompareOutput Then
      mcVBErr.WriteError errDiffFailed
   End If

   ' Delete the temporary files used to store the output of the compare and
   ' the errors generated by the compare
   Kill sDiffOutput
   Kill sCmpOutput

   Exit Function

CompareOutputErr:
   mcVBErr.LogVBErrors
   CompareOutput = False

End Function
Public Property Get InstanceId() As Long

   InstanceId = mlngInstanceId

End Property
Public Property Let InstanceId(ByVal vdata As Long)

   mlngInstanceId = vdata

End Property

Private Function ExecuteConstraint(vntConstraints As Variant, _
      ByRef intLoopIndex As Integer) As Boolean

   ' Returns True if there is a constraint in the passed in
   ' array that remains to be executed

   If IsArray(vntConstraints) And Not IsEmpty(vntConstraints) Then
      ExecuteConstraint = (LBound(vntConstraints) <= intLoopIndex) And
(intLoopIndex <= UBound(vntConstraints))
   Else
      ExecuteConstraint = False
   End If

End Function
Private Function NextStep() As cStep

   ' Determines which is the next step to be executed - it could
   ' be either a pre-execution step, the worker step itself
   ' or a post-execution step

   Dim cConsRec As cConstraint
   Dim cNextStepRec As cStep
   Dim vntStepConstraints As Variant

   ' Static variable to remember exactly where we are in the
   ' processing
   Static intIndex As Integer
   Static intNextStepType As NextNodeType

   On Error GoTo NextStepErr
```

```
  If mblnAbort = True Then
    ' The user has aborted the run - do not run any more
    ' processes for the step
    Set NextStep = Nothing
    Exit Function
  End If

  If intNextStepType = 0 Then
    ' First time through this function - set the Index and
    ' node type to initial values
    intNextStepType = mintWspPreConstraint
    intIndex = 0
    RaiseEvent StepStart(mcStep, Determine64BitTime(), mlngInstanceId)
  End If

  Do
    Select Case intNextStepType
      Case mintWspPreConstraint
        vntStepConstraints = mcvntWspPreCons

      Case mintPreConstraint
        vntStepConstraints = mcvntPreCons

      Case mintStep
        ' CONS:
        If mcStep.StepType = gintWorkerStep Then
          Set cNextStepRec = mcStep
        End If

      Case mintWspPostConstraint
        vntStepConstraints = mcvntWspPostCons

      Case mintPostConstraint
        vntStepConstraints = mcvntPostCons

    End Select

    If intNextStepType <> mintStep Then
      ' Check if there is a constraint to be executed
      If ExecuteConstraint(vntStepConstraints, intIndex) Then
        ' Get the corresponding step record to be executed
        ' Query the global step record for the current
        ' constraint
        Set cConsRec = vntStepConstraints(intIndex)

        Set cNextStepRec = mcGlobals.QueryStep(cConsRec.GlobalStepId)
        intIndex = intIndex + 1
      Else
        If intNextStepType = mintPostConstraint Then
          ' No more stuff to be executed for the step
          ' Raise a Done event
          Set cNextStepRec = Nothing

          ' Set the next step type to an invalid value
          intNextStepType = -1
        Else
          Call NextType(intNextStepType, intIndex)
        End If
      End If
    Else
      ' Increment the step type so we look at the post-
      ' execution steps the next time through
      Call NextType(intNextStepType, intIndex)
    End If

  Loop Until (Not cNextStepRec Is Nothing) Or _
      intNextStepType = -1

  Set NextStep = cNextStepRec

  Exit Function

NextStepErr:
  ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    mstrSource = mstrModuleName & "NextStep"
    Err.Raise vbObjectError + errNextStepFailed, mstrSource, _
      LoadResString(errNextStepFailed)

End Function
Public Sub Execute()
  ' This procedure is the method that executes the step that
  ' is assigned to the ExecuteStep property. It call a procedure
  ' to determine the next step to be executed.
  ' Then it initializes all the properties of the cExecuteSM object
  ' and calls it's run method to execute it.
  Dim cConn As cConnection
  Dim cRunConnDtl As cConnDtl

  On Error GoTo ExecuteErr

  ' If this procedure is called after a step has completed,
  ' we would have to check if we created any temporary files
  ' while executing that step
  If Not mcExecStep Is Nothing Then
    If Not StringEmpty(mcExecStep.StepTextFile) Or
mcExecStep.ExecutionMechanism = gintExecuteShell Then
      ' Remove the temporary file that we created while
      ' running this command
      Kill mstrCommand
    End If

    Call StepCompleted

    ' The VB errors class stores a reference to the Execute class since it uses
    ' a method of the class to write errors to the error log. Hence,
    ' release all references to the Execute object before destroying it.
    Set mcVBErr.ErrorFile = Nothing
    Set mcExecObj = Nothing

    ' Delete empty output and error files (generated by shell commands)
    ' (Can be done only after cleaning up cExecObj)
    Call DeleteEmptyOutputFiles
  Else
    ' First time through - initialize the location of output files
    msOutputDir = GetDefaultDir(mcStep.WorkspaceId, WspParameters)
    msOutputDir = msOutputDir & gstrFileSeparator & Trim(Str(RunId)) &
gstrFileSeparator
    ' Dummy file since the function expects a file name
    MakePathValid (msOutputDir & "a.txt")
  End If

  ' Call a procedure to determine the next step to be executed
  ' - could be a constraint or the step itself
  ' Initialize a module-level variable to the step being
  ' executed
  Set mcExecStep = NextStep
  If mcExecStep Is Nothing Then
    RaiseEvent StepComplete(mcStep, Determine64BitTime(), mlngInstanceId,
miStatus)
    ' No more stuff to execute
    Exit Sub
  End If

  Dim sStartDir As String

  Set mcExecObj = New EXECUTEDLLLib.Execute

  ' The VB errors class uses the WriteError method of the Execute class to write
  ' all VB errors to the error file for the step (this prevents a clash when the
  ' VB errors and Execution errors have to be written to the same log). Hence,
store
  ' a reference to the Execute object in mcVBErr
  msErrorFile = GetOutputFile(gsErrorFileSuffix)
  mcExecObj.ErrorFile = msErrorFile
  Call DeleteFile(msErrorFile, bCheckIfEmpty:=False)
  Set mcVBErr.ErrorFile = mcExecObj
```

```
   If mcExecStep.ExecutionMechanism = gintExecuteShell Then
      sStartDir = Trim$(GetShortName(SubstituteParameters( _
               mcExecStep.StartDir, mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters)))
      ' Dummy connection object
      Set cConn = New cConnection
      Set cRunConnDtl = New cConnDtl
   Else
      ' Find the connection string value and substitute parameter values in it
      Set cRunConnDtl =
WspConnDtls.GetConnectionDtl(mcExecStep.WorkspaceId,
mcExecStep.StartDir)
      Set cConn = WspConnections.GetConnection(mcExecStep.WorkspaceId,
cRunConnDtl.ConnectionString)
      sStartDir = Trim$(SubstituteParameters(cConn.ConnectionValue, _
            mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters))
   End If

   msOutputFile = GetOutputFile(gsOutputFileSuffix)
   Call DeleteFile(msOutputFile, bCheckIfEmpty:=False)
   mcExecObj.OutputFile = msOutputFile
'    mcExecObj.LogFile = GetShortName(SubstituteParameters( _
'          mcExecStep.LogFile, mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters))
   If mcExecStep.ExecutionMechanism = gintExecuteODBC And _
         cRunConnDtl.ConnType = ConnTypeDynamic Then
      Call mcExecObj.DoExecute(BuildCommandString(), sStartDir,
mcExecStep.ExecutionMechanism, _
            cConn.NoCountDisplay, cConn.NoExecute, cConn.ParseQueryOnly,
cConn.QuotedIdentifiers, _
            cConn.AnsiNulls, cConn.ShowQueryPlan, cConn.ShowStatsTime,
cConn.ShowStatsIO, _
            cConn.RowCount, cConn.QueryTimeOut, gstrEmptyString)
   Else
      Call mcExecObj.DoExecute(BuildCommandString(), sStartDir,
mcExecStep.ExecutionMechanism, _
            cConn.NoCountDisplay, cConn.NoExecute, cConn.ParseQueryOnly,
cConn.QuotedIdentifiers, _
            cConn.AnsiNulls, cConn.ShowQueryPlan, cConn.ShowStatsTime,
cConn.ShowStatsIO, _
            cConn.RowCount, cConn.QueryTimeOut, mcExecStep.StartDir)
   End If

   Exit Sub

ExecuteErr:
   Call HandleExecutionError

   ' We can assume that if we are in this function, a StepStart event has been
triggered already.
   RaiseEvent StepComplete(mcStep, Determine64BitTime(), mlngInstanceId,
miStatus)

End Sub
Private Function BuildCommandString() As String
   ' Process text to be executed - either from the text
   ' field or read it from a file.
   ' This function will always return the command text for ODBC commands
   ' and a file name for Shell commands
   Dim sFile As String
   Dim sCommand As String
   Dim sTemp As String

   On Error GoTo BuildCommandStringErr

   If Not StringEmpty(mcExecStep.StepTextFile) Then
      ' Substitute parameter values and environment variables
      ' in the filename
      msRunStepDtl = SubstituteParameters(mcExecStep.StepTextFile, _
            mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters)
      sFile = GetShortName(msRunStepDtl)

      mstrCommand = SubstituteParametersInText(sFile,
mcExecStep.WorkspaceId)

      If mcExecStep.ExecutionMechanism = gintExecuteODBC Then
         ' Read the contents of the file and pass it to ODBC
         BuildCommandString = ReadCommandFromFile(mstrCommand)
      Else
         BuildCommandString = mstrCommand
      End If
   Else
      ' Substitute parameter values and environment variables
      ' in the step text
      msRunStepDtl = SubstituteParameters(mcExecStep.StepText, _
            mcExecStep.WorkspaceId, mcIterators,
WspParameters:=WspParameters)
      mstrCommand = msRunStepDtl

      If mcExecStep.ExecutionMechanism = gintExecuteShell Then
         ' Write the command to a temp file (enables us to execute multiple
         ' commands via the command interpreter)
         mstrCommand = WriteCommandToFile(msRunStepDtl)
         BuildCommandString = mstrCommand
      Else
         BuildCommandString = SQLFixup(msRunStepDtl)
      End If
   End If

   If CreateInputFiles Then
      Call CreateStepTextFile
   End If

   Exit Function

BuildCommandStringErr:
   ' Log the error code raised by the Execute procedure
'    Call LogErrors(Errors)
   mcVBErr.LogVBErrors

   On Error GoTo 0
   mstrSource = mstrModuleName & "Execute"
   Err.Raise vbObjectError + errExecuteStepFailed, mstrSource, _
      LoadResString(errExecuteStepFailed) & mstrCommand

End Function
Public Sub Abort()

   On Error GoTo AbortErr

   ' Setting the Abort flag to True will ensure that we
   ' don't execute any more processes for this step
   mblnAbort = True

   If Not mcExecObj Is Nothing Then
      mcExecObj.Abort
   Else
      ' We are not in the middle of execution yet
   End If

   Exit Sub

AbortErr:
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errProgramError, _
      mstrModuleName & "Abort", _
      LoadResString(errProgramError)

End Sub
Private Sub NextType(ByRef StepType As NextNodeType, _
      ByRef Position As Integer)

   StepType = StepType + 1
```

```
   Position = 0

End Sub
Private Sub StepCompleted()

   On Error GoTo StepCompletedErr

   If Not mcExecStep Is Nothing Then
      If mcExecStep Is mcStep Then
         miStatus = InitializeExecStatus
         If miStatus = gintFailed Then
            ' Create input files if the step failed execution and one hasn't been
created already
            If Not CreateInputFiles Then CreateStepTextFile
            Call mcVBErr.WriteError(errExecuteStepFailed, _
                  OptArgs:="Continuation criteria for the step is: " &
gsContCriteria(mcStep.ContinuationCriteria))
         End If
      End If
   End If

   Exit Sub

StepCompletedErr:
   ' Log the error code raised by Visual Basic
   miStatus = gintFailed
   mcVBErr.LogVBErrors
   Call mcVBErr.WriteError(errExecuteStepFailed, _
         OptArgs:="Continuation criteria for the step is: " &
gsContCriteria(mcStep.ContinuationCriteria))

End Sub
Private Sub DeleteEmptyOutputFiles()

   On Error GoTo DeleteEmptyOutputFilesErr

   ' Delete empty output and error files
   If Not mcExecStep Is Nothing Then
      Call DeleteFile(msErrorFile, bCheckIfEmpty:=True)
      Call DeleteFile(msOutputFile, bCheckIfEmpty:=True)
   End If

   Exit Sub

DeleteEmptyOutputFilesErr:
   ' Not a critical error - continue

End Sub
Private Function ReadCommandFromFile(strFileName As String) As String

   ' Returns the contents of the passed in file

   Dim sCommand As String
   Dim sTemp As String
   Dim InputFile As Integer

   On Error GoTo ReadCommandFromFileErr

   If Not StringEmpty(strFileName) Then

      InputFile = FreeFile
      Open strFileName For Input Access Read As InputFile

      Line Input #InputFile, sCommand ' Read line into variable.

      Do While Not EOF(InputFile) ' Loop until end of file.
         Line Input #InputFile, sTemp ' Read line into variable.
         sCommand = sCommand & vbCrLf & sTemp
      Loop

      Close InputFile
   End If

   ReadCommandFromFile = sCommand

   Exit Function

ReadCommandFromFileErr:

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

End Function
Private Function SubstituteParametersIfPossible(strLabel As String)

   On Error GoTo SubstituteParametersIfPossibleErr

   SubstituteParametersIfPossible = SubstituteParameters(strLabel, _
         mcExecStep.WorkspaceId, mcIterators, WspParameters:=WspParameters)
   Exit Function

SubstituteParametersIfPossibleErr:
   SubstituteParametersIfPossible = strLabel

End Function
Private Function SubstituteParametersInText(strFileName As String, _
      lngWorkspace As Long) As String

   ' Reads each line in the passed in file, substitutes parameter
   ' values in the line and writes out the modified line to a
   ' temporary file that we create. The temporary file will be
   ' removed once the step completes execution.
   ' Returns the name of the newly created temporary file.

   Dim strTempFile As String
   Dim strTemp As String
   Dim strOutput As String
   Dim InputFile As Integer
   Dim OutputFile As Integer

   On Error GoTo SubstituteParametersInTextErr

   strTempFile = CreateTempFile()

   If Not StringEmpty(strFileName) Then

      InputFile = FreeFile
      Open strFileName For Input Access Read As InputFile

      OutputFile = FreeFile
      Open strTempFile For Output Access Write As OutputFile

      Do While Not EOF(InputFile) ' Loop until end of file.
         Line Input #InputFile, strTemp ' Read line into variable.
         strOutput = SubstituteParameters(strTemp, lngWorkspace, mcIterators,
WspParameters:=WspParameters)

         If mcExecStep.ExecutionMechanism = gintExecuteODBC Then strOutput
= SQLFixup(strOutput)

         Print #OutputFile, strOutput
         BugMessage strOutput
      Loop

   End If

   Close InputFile
   Close OutputFile

   SubstituteParametersInText = strTempFile

   Exit Function
```

```
SubstituteParametersInTextErr:

    ' Log the error code raised by Visual Basic
'    Call LogErrors(Errors)
    mcVBErr.LogVBErrors
    mstrSource = mstrModuleName & "SubstituteParametersInText"
    On Error GoTo 0
    Err.Raise vbObjectError + errSubValuesFailed, _
            gstrSource, _
            LoadResString(errSubValuesFailed)

End Function

Private Function WriteCommandToFile(sCommand As String, Optional sFile As
String = gstrEmptyString) As String

    ' Writes the command text to a temporary file
    ' Returns the name of the temporary file

    Dim OutputFile As Integer

    On Error GoTo WriteCommandToFileErr

    If StringEmpty(sFile) Then
        sFile = CreateTempFile()
    End If

    OutputFile = FreeFile
    Open sFile For Output Access Write As OutputFile

    Print #OutputFile, sCommand

    Close OutputFile

    WriteCommandToFile = sFile

    Exit Function

WriteCommandToFileErr:

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

End Function


Public Property Get WspPreCons() As Variant
    WspPreCons = mcvntWspPreCons
End Property
Public Property Let WspPreCons(ByVal vdata As Variant)
    mcvntWspPreCons = vdata
End Property

Public Property Get WspPostCons() As Variant
    WspPostCons = mcvntWspPostCons
End Property
Public Property Let WspPostCons(ByVal vdata As Variant)
    mcvntWspPostCons = vdata
End Property

Public Property Get PreCons() As Variant
    PreCons = mcvntPreCons
End Property
Public Property Let PreCons(ByVal vdata As Variant)
    mcvntPreCons = vdata
End Property

Public Property Get PostCons() As Variant
    PostCons = mcvntPostCons
End Property
Public Property Let PostCons(ByVal vdata As Variant)
    mcvntPostCons = vdata
End Property

Public Property Set Globals(cRunSteps As cArrSteps)

    Set mcGlobals = cRunSteps

End Property
Public Property Set ExecuteStep(cRunStep As cStep)

    Set mcStep = cRunStep

End Property
Public Property Get Globals() As cArrSteps

    Set Globals = mcGlobals

End Property
Public Property Get ExecuteStep() As cStep

    Set ExecuteStep = mcStep

End Property
Public Property Set Iterators(vdata As cRunColIt)

    Set mcIterators = vdata

End Property
Private Sub Class_Initialize()

    ' Initialize the Abort flag to False
    mblnAbort = False
    Set mcVBErr = New cVBErrorsSM
    Set mcTermProcess = New cTermProcess

End Sub

Private Sub Class_Terminate()

    On Error GoTo Class_TerminateErr

    Set mcExecObj = Nothing
    Set mcVBErr = Nothing
    Set mcTermProcess = Nothing

    Exit Sub

Class_TerminateErr:
    Call LogErrors(Errors)

End Sub

Private Sub mcExecObj_Start(ByVal StartTime As Currency)
    ' Raise an event indicating that the step has begun execution
    RaiseEvent ProcessStart(mcExecStep, msRunStepDtl, StartTime,
mlngInstanceId)
End Sub

Private Sub mcExecObj_Complete(ByVal EndTime As Currency, ByVal Elapsed
As Long)

    On Error GoTo mcExecObj_CompleteErr

    Debug.Print Elapsed
    RaiseEvent ProcessComplete(mcExecStep, EndTime, mlngInstanceId, Elapsed)
    mcTermProcess.ProcessTerminated

    Exit Sub

mcExecObj_CompleteErr:
    Call LogErrors(Errors)
```

```
End Sub

Private Sub mcTermProcess_TermProcessExists()

   On Error GoTo TermProcessExistsErr

   ' Call a procedure to execute the next step, if any
   Call Execute

   Exit Sub

TermProcessExistsErr:
   ' Log the error code raised by the Execute procedure
   Call LogErrors(Errors)

End Sub
```

cRunWorkspace.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cRunWorkspace"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cRunWorkspace.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This class loads all the information necessary to
'              execute a workspace and calls cRunInst to execute the workspace.
'              It also propagates Step start and complete and
'              Run start and complete events.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "cRunWorkspace."
Private mstrSource As String

Private mcRunSteps As cArrSteps
Private mcRunParams As cArrParameters
Private mcRunConstraints As cArrConstraints
Private mcRunConnections As cConnections
Private mcRunConnDtls As cConnDtls
Private mcvntWspPreCons As Variant
Private mcvntWspPostCons As Variant
Private mdbsLoadDb As Database
Private mlngRunId As Long
Private mlngWorkspaceId As Long
Private mField As cStringSM
Public CreateInputFiles As Boolean

Private WithEvents mcRun As cRunInst
Attribute mcRun.VB_VarHelpID = -1

Public Event RunStart(dtmStartTime As Currency, strWspLog As String, lRunId
As Long)
Public Event RunComplete(dtmEndTime As Currency)
Public Event StepStart(cStepRecord As cStep, dtmStartTime As Currency,
lngInstanceId As Long, _
      sPath As String, sIts As String)
Public Event StepComplete(cStepRecord As cStep, dtmEndTime As Currency,
lngInstanceId As Long)
Public Event ProcessStart(cStepRecord As cStep, strCommand As String, _
      dtmStartTime As Currency, lngInstanceId As Long)
Public Event ProcessComplete(cStepRecord As cStep, dtmEndTime As Currency,
lngInstanceId As Long)
Public Function InstancesForStep(lngStepId As Long, iStatus As InstanceStatus)
As cInstances
   ' Returns an array of all the instances for a step

   If mcRun Is Nothing Then
      Set InstancesForStep = Nothing
   Else
      Set InstancesForStep = mcRun.InstancesForStep(lngStepId, iStatus)
   End If

End Function
Private Sub InsertRunDetail(cStepRecord As cStep, _
      strCommand As String, dtmStartTime As Currency, _
      lngInstanceId As Long, lParentInstanceId As Long, sItValue As String)
   ' Inserts a new run detail record into the database

   Dim strInsert As String
   Dim qy As QueryDef

   On Error GoTo InsertRunDetailErr
   mstrSource = mstrModuleName & "InsertRunDetail"

   strInsert = "insert into run_step_details " & _
         "( run_id, step_id, version_no, instance_id, parent_instance_id, " & _
         " command, start_time, iterator_value ) " & _
         " values ( "

#If USE_JET Then

   strInsert = strInsert & " [r_id], [s_id], [ver_no], [i_id], [p_i_id], " & _
         " [com], [s_date], [it_val] )"

   Set qy = mdbsLoadDb.CreateQueryDef( _
         gstrEmptyString, strInsert)

   ' Call a procedure to assign the Querydef parameters
   Call AssignParameters(qy, StartTime:=dtmStartTime, _
         StepId:=cStepRecord.StepId, _
         Version:=cStepRecord.VersionNo, _
         InstanceId:=lngInstanceId, _
         Command:=strCommand)

   qy.Execute dbFailOnError
   qy.Close

#Else

   strInsert = strInsert & Str(mlngRunId) _
      & ", " & Str(cStepRecord.StepId) _
      & ", " & mField.MakeStringFieldValid(cStepRecord.VersionNo) _
      & ", " & Str(lngInstanceId) _
      & ", " & Str(lParentInstanceId) _
      & ", " & mField.MakeStringFieldValid(strCommand) _
      & ", " & Str(dtmStartTime) _
      & ", " & mField.MakeStringFieldValid(sItValue)

   strInsert = strInsert & " ) "

   mdbsLoadDb.Execute strInsert, dbFailOnError

#End If

   Exit Sub

InsertRunDetailErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "InsertRunDetail"
   On Error GoTo 0
   Err.Raise vbObjectError + errUpdateRunDataFailed, _
         mstrSource, _
         LoadResString(errUpdateRunDataFailed)
```

```
End Sub
Private Sub UpdateRunDetail(cStepRecord As cStep, _
      dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)
   ' Updates the run detail record in the database

   Dim strUpdate As String
   Dim qy As QueryDef

   On Error GoTo UpdateRunDetailErr

   strUpdate = "update run_step_details " & _
         " set end_time = [e_date], elapsed_time = [elapsed] " & _
         " where run_id = [r_id] " & _
         " and step_id = [s_id] " & _
         " and version_no = [ver_no] " & _
         " and instance_id = [i_id] "

   Set qy = mdbsLoadDb.CreateQueryDef( _
         gstrEmptyString, strUpdate)

   ' Call a procedure to assign the Querydef parameters
   Call AssignParameters(qy, EndTime:=dtmEndTime, _
         StepId:=cStepRecord.StepId, _
         Version:=cStepRecord.VersionNo, _
         InstanceId:=lngInstanceId, Elapsed:=lElapsed)

   qy.Execute dbFailOnError
   qy.Close

   Exit Sub

UpdateRunDetailErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "UpdateRunDetail"
   On Error GoTo 0
   Err.Raise vbObjectError + errUpdateRunDataFailed, _
         mstrSource, _
         LoadResString(errUpdateRunDataFailed)

End Sub
Private Function InsertRunHeader(dtmStartTime As Currency) As Long
   ' Inserts a new run header record into the database
   ' and returns the id for the run

   Dim strInsert As String
   Dim qy As QueryDef

   On Error GoTo InsertRunHeaderErr

   strInsert = "insert into run_header " & _
      "( run_id, workspace_id, start_time ) " & _
      " values ( " & _
      " [r_id], [w_id], [s_date] )"

   Set qy = mdbsLoadDb.CreateQueryDef( _
         gstrEmptyString, strInsert)

   ' Call a procedure to execute the Querydef object
   Call AssignParameters(qy, StartTime:=dtmStartTime)

   qy.Execute dbFailOnError
   qy.Close

   InsertRunHeader = mlngRunId
   Exit Function

InsertRunHeaderErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "InsertRunHeader"
   On Error GoTo 0
   Err.Raise vbObjectError + errUpdateRunDataFailed, _
         mstrSource, _
         LoadResString(errUpdateRunDataFailed)

End Function
Private Sub InsertRunParameters(dtmStartTime As Currency)
   ' Inserts a new run header record into the database
   ' and returns the id for the run

   Dim strInsert As String
   Dim qy As QueryDef
   Dim cParamRec As cParameter
   Dim lngIndex As Long

   On Error GoTo InsertRunParametersErr

   strInsert = "insert into run_parameters " & _
      "( run_id, parameter_name, parameter_value ) " & _
      " values ( " & _
      " [r_id], [p_name], [p_value] )"

   Set qy = mdbsLoadDb.CreateQueryDef( _
         gstrEmptyString, strInsert)
   qy.Parameters("r_id").Value = mlngRunId

   For lngIndex = 0 To mcRunParams.ParameterCount - 1
      Set cParamRec = mcRunParams(lngIndex)

      qy.Parameters("p_name").Value = cParamRec.ParameterName
      qy.Parameters("p_value").Value = cParamRec.ParameterValue
      qy.Execute dbFailOnError

   Next lngIndex

   qy.Close

   Exit Sub

InsertRunParametersErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "InsertRunParameters"
   On Error GoTo 0
   Err.Raise vbObjectError + errUpdateRunDataFailed, _
         mstrSource, _
         LoadResString(errUpdateRunDataFailed)

End Sub
Private Sub AssignParameters(qyExec As DAO.QueryDef, _
      Optional StartTime As Currency = 0, _
      Optional EndTime As Currency = 0, _
      Optional StepId As Long = 0, _
      Optional Version As String = gstrEmptyString, _
      Optional InstanceId As Long = 0, _
      Optional ParentInstanceId As Long = 0, _
      Optional Command As String = gstrEmptyString, _
      Optional Elapsed As Long = 0, _
      Optional ItValue As String = gstrEmptyString)
   ' Assigns values to the parameters in the querydef object

   Dim prmParam As DAO.Parameter

   On Error GoTo AssignParametersErr
   mstrSource = mstrModuleName & "AssignParameters"

   For Each prmParam In qyExec.Parameters
      Select Case prmParam.Name
         Case "[w_id]"
            prmParam.Value = mlngWorkspaceId

         Case "[r_id]"
            prmParam.Value = mlngRunId

         Case "[s_id]"
            BugAssert StepId <> 0
            prmParam.Value = StepId

         Case "[ver_no]"
            BugAssert Not StringEmpty(Version)
```

```
            prmParam.Value = Version

        Case "[i_id]"
            BugAssert InstanceId <> 0
            prmParam.Value = InstanceId

        Case "[p_i_id]"
            prmParam.Value = ParentInstanceId

        Case "[com]"
            BugAssert Not StringEmpty(Command)
            prmParam.Value = Command

        Case "[s_date]"
            BugAssert StartTime <> 0
            prmParam.Value = StartTime

        Case "[e_date]"
            BugAssert EndTime <> 0
            prmParam.Value = EndTime

        Case "[elapsed]"
            prmParam.Value = Elapsed

        Case "[it_val]"
            prmParam.Value = ItValue

        Case Else
            ' Write the parameter name that is faulty
            WriteError errInvalidParameter, mstrSource, _
                prmParam.Name
            On Error GoTo 0
            Err.Raise errInvalidParameter, mstrSource, _
                LoadResString(errInvalidParameter)
    End Select
  Next prmParam

  Exit Sub

AssignParametersErr:

  mstrSource = mstrModuleName & "AssignParameters"
  Call LogErrors(Errors)
  On Error GoTo 0
  Err.Raise vbObjectError + errAssignParametersFailed, _
      mstrSource, LoadResString(errAssignParametersFailed)

End Sub
Private Sub RunStartProcessing(dtmStartTime As Currency)

  On Error GoTo RunStartProcessingErr

  ' Insert the run header into the database
  Call InsertRunHeader(dtmStartTime)

  ' Insert the run parameters into the database
  Call InsertRunParameters(dtmStartTime)

  Exit Sub

RunStartProcessingErr:
  ' Log the error code raised by Visual Basic
  Call LogErrors(Errors)
  mstrSource = mstrModuleName & "RunStartProcessing"
  ShowError errUpdateRunDataFailed
  WriteError errUpdateRunDataFailed, mstrSource

End Sub
Private Sub ProcessStartProcessing(cStepRecord As cStep, _
    strCommand As String, dtmStartTime As Currency, lngInstanceId As Long,
_
    lParentInstanceId As Long, sItValue As String)

  On Error GoTo ProcessStartProcessingErr

  ' Insert the run detail into the database
  Call InsertRunDetail(cStepRecord, strCommand, dtmStartTime, lngInstanceId,
_
      lParentInstanceId, sItValue)

  Exit Sub

ProcessStartProcessingErr:
  ' Log the error code raised by Visual Basic
  Call LogErrors(Errors)
  mstrSource = mstrModuleName & "ProcessStartProcessing"
  ShowError errUpdateRunDataFailed
  WriteError errUpdateRunDataFailed, mstrSource

End Sub
Private Sub StepStartProcessing(cStepRecord As cStep, dtmStartTime As
Currency, _
    lngInstanceId As Long, lParentInstanceId As Long, sItValue As String)

  On Error GoTo StepStartProcessingErr

  ' Since ProcessStart events won't be triggered for manager steps
  If cStepRecord.StepType = gintManagerStep Then
    ' Insert the run detail into the database
    Call InsertRunDetail(cStepRecord, cStepRecord.StepLabel, _
        dtmStartTime, lngInstanceId, lParentInstanceId, sItValue)
  End If

  Exit Sub

StepStartProcessingErr:
  ' Log the error code raised by Visual Basic
  Call LogErrors(Errors)
  mstrSource = mstrModuleName & "StepStartProcessing"
  ShowError errUpdateRunDataFailed

End Sub
Private Sub ProcessCompleteProcessing(cStepRecord As cStep, _
    dtmStartTime As Currency, lngInstanceId As Long, lElapsed As Long)

  On Error GoTo ProcessCompleteProcessingErr

  ' Insert the run detail into the database
  Call UpdateRunDetail(cStepRecord, dtmStartTime, lngInstanceId, lElapsed)

  Exit Sub

ProcessCompleteProcessingErr:
  ' Log the error code raised by Visual Basic
  Call LogErrors(Errors)
  mstrSource = mstrModuleName & "ProcessCompleteProcessing"
  ShowError errUpdateRunDataFailed

End Sub
Private Sub StepCompleteProcessing(cStepRecord As cStep, _
    dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

  On Error GoTo StepCompleteProcessingErr

  ' Since ProcessComplete events won't be triggered for manager steps
  If cStepRecord.StepType = gintManagerStep Then
    ' Update the run detail in the database
    Call UpdateRunDetail(cStepRecord, dtmEndTime, lngInstanceId, lElapsed)
  End If

  Exit Sub

StepCompleteProcessingErr:
  ' Log the error code raised by Visual Basic
  Call LogErrors(Errors)
  ShowError errUpdateRunDataFailed

End Sub
```

```
Private Sub RunCompleteProcessing(dtmEndTime As Currency)

   On Error GoTo RunCompleteProcessingErr

   ' Update the header record with the end time for the run
   Call UpdateRunHeader(dtmEndTime)

   Exit Sub

RunCompleteProcessingErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   ShowError errUpdateRunDataFailed

End Sub

Private Sub UpdateRunHeader(ByVal dtmEndTime As Currency)
   ' Updates the run header record with the end date

   Dim strUpdate As String
   Dim qy As QueryDef

   On Error GoTo UpdateRunHeaderErr

   strUpdate = "update run_header " & _
         " set end_time = [e_date] " & _
         " where run_id = [r_id] "

   Set qy = mdbsLoadDb.CreateQueryDef( _
         gstrEmptyString, strUpdate)

   ' Call a procedure to execute the Querydef object
   Call AssignParameters(qy, EndTime:=dtmEndTime)

   qy.Execute dbFailOnError
   qy.Close

   Exit Sub

UpdateRunHeaderErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "UpdateRunHeader"
   On Error GoTo 0
   Err.Raise vbObjectError + errUpdateRunDataFailed, _
         mstrSource, _
         LoadResString(errUpdateRunDataFailed)

End Sub

Public Property Let WorkspaceId(ByVal vdata As Long)
   mlngWorkspaceId = vdata
End Property
Public Property Get WorkspaceId() As Long
   WorkspaceId = mlngWorkspaceId
End Property
Public Sub RunWorkspace()

   Dim cRunSeq As cSequence

   On Error GoTo RunWorkspaceErr

   ' Call a procedure to load the module-level structures
   ' with all the step and parameter data for the run
   If LoadRunData = False Then
      ' Error handled by the function already
      Exit Sub
   End If

   ' Retrieve the next run identifier using the sequence class
   Set cRunSeq = New cSequence
   Set cRunSeq.IdDatabase = dbsAttTool
   cRunSeq.IdentifierColumn = "run_id"
   mlngRunId = cRunSeq.Identifier
   Set cRunSeq = Nothing

   Call mcRunParams.InitBuiltInsForRun(mlngWorkspaceId, mlngRunId)

   Set mcRun.Constraints = mcRunConstraints
   mcRun.WspPreExecution = mcvntWspPreCons
   mcRun.WspPostExecution = mcvntWspPostCons

   Set mcRun.Steps = mcRunSteps
   Set mcRun.Parameters = mcRunParams
   Set mcRun.RunConnections = mcRunConnections
   Set mcRun.RunConnDtls = mcRunConnDtls

   mcRun.WspId = mlngWorkspaceId
   mcRun.RootKey = LabelStep(mlngWorkspaceId)
   mcRun.RunId = mlngRunId
   mcRun.CreateInputFiles = CreateInputFiles

   mcRun.Run

   Exit Sub

RunWorkspaceErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)

End Sub
Public Property Get LoadDb() As Database

   Set LoadDb = mdbsLoadDb

End Property
Public Property Set LoadDb(vdata As Database)

   Set mdbsLoadDb = vdata

End Property
Private Function LoadRunData() As Boolean

   ' Loads the step, parameter and constraint arrays
   ' with all the data for the workspace. Returns False
   ' if a failure occurs

   Dim strWorkspaceName As String
   Dim recWspSteps As Recordset
   Dim qySteps As DAO.QueryDef
   Dim recWspParams As Recordset
   Dim qyParams As DAO.QueryDef
   Dim recWspConns As Recordset
   Dim qyConns As DAO.QueryDef
   Dim recWspConnDtls As Recordset
   Dim qyConnDtls As DAO.QueryDef

   On Error GoTo LoadRunDataErr

   Set mcRunSteps.StepDB = mdbsLoadDb
   Set mcRunParams.ParamDatabase = mdbsLoadDb
   Set mcRunConstraints.ConstraintDB = mdbsLoadDb
   Set mcRunConnections.ConnDb = mdbsLoadDb
   Set mcRunConnDtls.ConnDb = mdbsLoadDb

   ' Read all the step and parameter data for the workspace
   Call ReadWorkspaceData(mlngWorkspaceId, mcRunSteps, _
         mcRunParams, mcRunConstraints, mcRunConnections, mcRunConnDtls,
_
         recWspSteps, qySteps, recWspParams, qyParams, recWspConns,
qyConns, _
         recWspConnDtls, qyConnDtls)

   ' Load all the pre- and post-execution constraints that
   ' have been defined for the workspace
   mcvntWspPreCons = mcRunConstraints.ConstraintsForWsp( _
      mlngWorkspaceId, _
      gintPreStep, _
      blnSort:=True, _
```

```
        blnGlobalConstraintsOnly:=True)
    mcvntWspPostCons = mcRunConstraints.ConstraintsForWsp( _
        mlngWorkspaceId, _
        gintPostStep, _
        blnSort:=True, _
        blnGlobalConstraintsOnly:=True)

    On Error Resume Next
    recWspSteps.Close
    qySteps.Close
    recWspParams.Close
    qyParams.Close
    recWspConns.Close
    qyConns.Close

    LoadRunData = True

    Exit Function

LoadRunDataErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ShowError errLoadRunDataFailed
    LoadRunData = False

End Function
Public Sub StopRun()

    On Error GoTo StopRunErr

    If mcRun Is Nothing Then
        ' We haven't been the run yet, so do nothing
    Else
        mcRun.StopRun
    End If

    Exit Sub

StopRunErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' Errors would have been displayed by the called process

End Sub
Public Sub AbortRun()

    On Error GoTo AbortRunErr

    If mcRun Is Nothing Then
        ' We haven't been the run yet, so do nothing
    Else
        mcRun.Abort
    End If

    Exit Sub

AbortRunErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' Errors would have been displayed by the called process

End Sub

Private Sub Class_Initialize()

    ' Create instances of the step, parameter and constraint arrays
    Set mcRunSteps = New cArrSteps
    Set mcRunParams = New cArrParameters
    Set mcRunConstraints = New cArrConstraints
    Set mcRunConnections = New cConnections
    Set mcRunConnDtls = New cConnDtls
    Set mcRun = New cRunInst
    Set mField = New cStringSM
End Sub
Private Sub Class_Terminate()

    On Error GoTo UnLoadRunDataErr

    ' Clears the step, parameter and constraint arrays
    Set mcRunSteps = Nothing
    Set mcRunParams = Nothing
    Set mcRunConstraints = Nothing
    Set mcRunConnections = Nothing
    Set mcRunConnDtls = Nothing

    Set mcRun = Nothing
    Set mdbsLoadDb = Nothing
    Set mField = Nothing

    Exit Sub

UnLoadRunDataErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' Not a critical error - continue
    Resume Next

End Sub

Private Sub mcRun_ProcessComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    RaiseEvent ProcessComplete(cStepRecord, dtmEndTime, lngInstanceId)
    Call ProcessCompleteProcessing(cStepRecord, dtmEndTime, lngInstanceId,
lElapsed)

End Sub

Private Sub mcRun_ProcessStart(cStepRecord As cStep, _
        strCommand As String, dtmStartTime As Currency, lngInstanceId As Long,
_
        lParentInstanceId As Long, sItValue As String)

    RaiseEvent ProcessStart(cStepRecord, strCommand, dtmStartTime,
lngInstanceId)
    Call ProcessStartProcessing(cStepRecord, strCommand, dtmStartTime,
lngInstanceId, _
            lParentInstanceId, sItValue)

End Sub

Private Sub mcRun_RunComplete(dtmEndTime As Currency)

    Debug.Print "Run ended at: " & CStr(dtmEndTime)
    Call RunCompleteProcessing(dtmEndTime)

    RaiseEvent RunComplete(dtmEndTime)

End Sub
Private Sub mcRun_RunStart(dtmStartTime As Currency, strWspLog As String)

    RaiseEvent RunStart(dtmStartTime, strWspLog, mlngRunId)
    Debug.Print "Run started at: " & CStr(dtmStartTime)

    Call RunStartProcessing(dtmStartTime)

End Sub
Private Sub mcRun_StepComplete(cStepRecord As cStep, _
        dtmEndTime As Currency, lngInstanceId As Long, lElapsed As Long)

    RaiseEvent StepComplete(cStepRecord, dtmEndTime, lngInstanceId)
'    BugMessage "Step: " & cStepRecord.StepLabel & " has completed!"

    Call StepCompleteProcessing(cStepRecord, dtmEndTime, lngInstanceId,
lElapsed)

End Sub
```

```
Private Sub mcRun_StepStart(cStepRecord As cStep, dtmStartTime As Currency,
_
      lngInstanceId As Long, lParentInstanceId As Long, sPath As String, sIts As
String, sItValue As String)

   RaiseEvent StepStart(cStepRecord, dtmStartTime, lngInstanceId, sPath, sIts)
   'bugmessage "Step: " & cStepRecord.StepLabel & " has started."

   Call StepStartProcessing(cStepRecord, dtmStartTime, lngInstanceId,
lParentInstanceId, sItValue)

End Sub
```

CSequence.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cSequence"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cSequence.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     This class uses the att_identifiers table to generate unique
'              identifiers.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mlngIdentifier As Long
Private mstrIdentifierColumn As String
Private mrecIdentifiers As Recordset
Private mdbsDatabase As Database

Private Const mstrEmptyString = ""

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cSequence."

Private Sub CreateIdRecord()
   ' Creates a record with all identifiers having an initial value of 1

   Dim sSql As String
   Dim pId As DAO.Parameter
   Dim qyId As DAO.QueryDef

   sSql = "insert into att_identifiers (" & _
          " workspace_id, parameter_id, step_id, " & _
          " constraint_id, run_id, connection_id " & _
          ", " & FLD_ID_CONN_NAME & _
          " ) values ( " & _
          "[w_id], [p_id], [s_id], [c_id], [r_id], [conn_id], [conn_dtl_id] )"
   Set qyId = mdbsDatabase.CreateQueryDef(gstrEmptyString, sSql)
   For Each pId In qyId.Parameters
      pId.Value = glMinId
   Next pId
   qyId.Execute dbFailOnError
   qyId.Close

End Sub
Private Sub CreateIdRecordset()

   Dim strSql As String

   ' Initialize the recordset with all identifiers
   strSql = "select * from att_identifiers"
   Set mrecIdentifiers = mdbsDatabase.OpenRecordset(strSql,
dbOpenForwardOnly)

   If mrecIdentifiers.RecordCount = 0 Then
      CreateIdRecord
      Set mrecIdentifiers = mdbsDatabase.OpenRecordset(strSql,
dbOpenForwardOnly)
   End If

   BugAssert mrecIdentifiers.RecordCount <> 0

End Sub

Public Property Set IdDatabase(vdata As Database)

   Set mdbsDatabase = vdata

End Property

Public Property Let IdentifierColumn(vdata As String)

   Dim intIndex As Integer

   On Error GoTo IdentifierColumnErr

   ' Initialize the return value to an empty string
   mstrIdentifierColumn = mstrEmptyString
   Call CreateIdRecordset

   For intIndex = 0 To mrecIdentifiers.Fields.Count - 1

      If LCase(Trim(mrecIdentifiers.Fields(intIndex).Name)) = _
            LCase(Trim(vdata)) Then

         ' Valid column name
         mstrIdentifierColumn = vdata
         Exit Property
      End If

   Next intIndex

   BugAssert True, "Invalid column name!"

   Exit Property

IdentifierColumnErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "IdentifierColumn"
   On Error GoTo 0
   Err.Raise vbObjectError + errIdentifierColumnFailed, _
         mstrSource, _
         LoadResString(errIdentifierColumnFailed)

End Property
Public Property Get Identifier() As Long
   Dim strSql As String

   On Error GoTo GetIdentifierErr

   BugAssert mstrIdentifierColumn <> mstrEmptyString

   ' Increment the identifier column by 1
   strSql = "update att_identifiers " & _
      " set " & mstrIdentifierColumn & _
      " = " & mstrIdentifierColumn & " + 1"
   mdbsDatabase.Execute strSql, dbFailOnError

   ' Refresh the recordset with identifier values
   Call CreateIdRecordset

   mlngIdentifier = mrecIdentifiers.Fields(mstrIdentifierColumn).Value
```

```
    Identifier = mlngIdentifier

    Exit Property

GetIdentifierErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Identifier"
    On Error GoTo 0
    Err.Raise vbObjectError + errGetIdentifierFailed, _
          mstrSource, _
          LoadResString(errGetIdentifierFailed)

End Property
Private Sub Class_Terminate()

    mrecIdentifiers.Close

End Sub
```

CStack.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cStack"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cStack.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     This class implements a stack of objects.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cStack."
Private mstrSource As String

Private mcVector As cVector
Private mlngCount As Long
Public Property Get Item(ByVal Position As Long) As Object
Attribute Item.VB_UserMemId = 0

    Set Item = mcVector(Position)

End Property

Public Sub Push(objToPush As Object)

    mcVector.Add objToPush

End Sub
Public Sub Clear()

    mcVector.Clear

End Sub

Public Function Pop() As Object

    If mcVector.Count > 0 Then
        Set Pop = mcVector.Delete(mcVector.Count - 1)
    Else
        Set Pop = Nothing
    End If

End Function
Public Function Count() As Long

    Count = mcVector.Count

End Function


Private Sub Class_Initialize()

    Set mcVector = New cVector

End Sub


Private Sub Class_Terminate()

    Set mcVector = Nothing

End Sub
```

CStep.cls

```
5:28 PM 11/15/2007VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Attribute VB_Ext_KEY = "SavedWithClassBuilder" ,"Yes"
Attribute VB_Ext_KEY = "Top_Level" ,"Yes"
'  FILE:       cStep.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     Encapsulates the properties and methods of a step.
'              Contains functions to insert, update and delete
'              att_steps records from the database.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Local variable(s) to hold property value(s)
Private mlngStepId As Long
Private mstrVersionNo As String
Private mstrStepLabel As String
Private mstrStepTextFile As String
Private mstrStepText As String
Private mstrStartDir As String
Private mlngWorkspaceId As Integer
Private mlngParentStepId As Integer
Private mstrParentVersionNo As String
Private mintSequenceNo As Integer
Private mintStepLevel As Integer
Private mblnEnabledFlag As Boolean
Private mstrDegreeParallelism As String
Private mintExecutionMechanism As Integer
Private mstrFailureDetails As String
Private mintContinuationCriteria As Integer
Private mblnGlobalFlag As Boolean
Private mblnArchivedFlag As Boolean
Private mstrOutputFile As String
'Private mstrLogFile As String
Private mstrErrorFile As String
Private mdbsDatabase As Database
Private mintStepType As Integer
Private mintOperation As Operation
```

```
Private mlngPosition As Long
Private mstrIteratorName As String
Private mcIterators As cNodeCollections
Private mbIsNewVersion As Boolean
Private msOldVersion As String

' The following constants are used throughout the project to
' indicate the different options selected by the user
' The options are presented to the user as control arrays of
' option buttons. These constants have to be in sync with the
' indexes of the option buttons.
' All the control arrays have an lbound of 1. The value 0 is
' used to indicate that the property being represented by the
' control array is not valid for the step
' Public enums are used since we cannot expose public constants
' in class modules. gintNoOption is applicable to all enums,
' but declared in the Execution method enum, since we cannot
' declare it more than once.

' Is here as a comment
' Has been defined in public.bas with the other object types
'Public Enum gintStepType
'    gintGlobalStep = 3
'    gintManagerStep
'    gintWorkerStep
'End Enum

' Execution Method options
Public Enum ExecutionMethod
    gintNoOption = 0
    gintExecuteODBC
    gintExecuteShell
End Enum

' Failure criteria options
Public Enum FailureCriteria
    gintFailureODBC = 1
    gintFailureTextCompare
End Enum

' Continuation criteria options
' Note: Update the initialization of gsContCriteria in Initialize() if the
' continuation criteria are modified
Public Enum ContinuationCriteria
    gintOnFailureAbort = 1
    gintOnFailureContinue
    gintOnFailureCompleteSiblings
    gintOnFailureAbortSiblings
    gintOnFailureSkipSiblings
    gintOnFailureAsk
End Enum

' The initial version #
Private Const mstrMinVersion As String = "0.0"

' End of constants for option button control arrays
' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cStep."

' The cSequence class is used to generate unique step identifiers
Private mStepSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM
Private Sub NewVersion()

mbIsNewVersion = True
msOldVersion = mstrVersionNo

End Sub
Public Function IsNewVersion() As Boolean
    IsNewVersion = mbIsNewVersion
End Function

Public Function OldVersionNo() As String
    OldVersionNo = msOldVersion
End Function

Public Sub SaveIterators()
    ' This procedure checks if any changes have been made
    ' to the iterators for the step. If so, it calls the
    ' methods of the iterator class to commit the changes
    Dim cItRec As cIterator
    Dim lngIndex As Long

    On Error GoTo SaveIteratorsErr

    For lngIndex = 0 To mcIterators.Count - 1
        Set cItRec = mcIterators(lngIndex)

        Select Case cItRec.IndOperation
            Case QueryOp
                ' No changes were made to the queried Step.
                ' Do nothing

            Case InsertOp
                cItRec.Add mlngStepId, mstrVersionNo
                cItRec.IndOperation = QueryOp

            Case UpdateOp
                cItRec.Update mlngStepId, mstrVersionNo
                cItRec.IndOperation = QueryOp

            Case DeleteOp
                cItRec.Delete mlngStepId, mstrVersionNo
                ' Remove the record from the collection
                mcIterators.Delete lngIndex

        End Select
    Next lngIndex

    Exit Sub

SaveIteratorsErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "SaveIterators"
    On Error GoTo 0
    Err.Raise vbObjectError + errSaveFailed, _
            mstrSource, _
            LoadResString(errSaveFailed)

End Sub
Public Property Get IndOperation() As Operation

    IndOperation = mintOperation

End Property
Public Property Let IndOperation(ByVal vdata As Operation)

    BugAssert vdata = QueryOp Or vdata = InsertOp Or vdata = UpdateOp Or
vdata = DeleteOp, "Invalid operation"
    mintOperation = vdata

End Property

Public Function Iterators() As Variant
    ' Returns a variant containing all the iterators that
    ' have been defined for the step

    Dim cStepIterators() As cIterator
    Dim cTempIt As cIterator
    Dim lngIndex As Long
    Dim lngItCount As Long

    On Error GoTo IteratorsErr
```

```
    lngItCount = 0
    For lngIndex = 0 To mcIterators.Count - 1
        ' Increase the array dimension and add the constraint
        ' to it
        Set cTempIt = mcIterators(lngIndex)

        If cTempIt.IndOperation <> DeleteOp Then
            ReDim Preserve cStepIterators(lngItCount)
            Set cStepIterators(lngItCount) = cTempIt
            lngItCount = lngItCount + 1
        End If

    Next lngIndex

    If lngItCount = 0 Then
        Iterators = Empty
    Else
        Iterators = cStepIterators()
    End If

    Call QuickSort(Iterators)

    Exit Function

IteratorsErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errIteratorsFailed, _
            mstrModuleName & "Iterators", _
            LoadResString(errIteratorsFailed)

End Function
Public Function IteratorCount() As Long
    ' Returns a count of all the iterators for the step

    Dim lngItCount As Long
    Dim lngIndex As Long
    Dim cTempIt As cIterator

    On Error GoTo IteratorsErr

    lngItCount = 0
    For lngIndex = 0 To mcIterators.Count - 1

        If mcIterators(lngIndex).IndOperation <> DeleteOp Then
            lngItCount = lngItCount + 1
        End If

    Next lngIndex

    IteratorCount = lngItCount

    Exit Function

IteratorsErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errIteratorsFailed, _
            mstrSource, _
            LoadResString(errIteratorsFailed)

End Function
Public Sub Validate()
    ' Each distinct object will have a Validate method which
    ' will check if the class properties are valid. This method
    ' will be used to check interdependant properties that
    ' cannot be validated by the let procedures.
    ' It should be called by the add and modify methods of the class

    ' Check if the step label has been specified
    If StringEmpty(mstrStepLabel) Then
        ShowError errStepLabelMandatory
        On Error GoTo 0
        Err.Raise vbObjectError + errValidateFailed, _
                "Validate", LoadResString(errValidateFailed)
    End If

    If Not IsStringEmpty(mstrStepText) And Not IsStringEmpty(mstrStepTextFile)
Then
        ShowError errStepTextOrFile
        On Error GoTo 0
        Err.Raise vbObjectError + errStepTextOrFile, _
            "Validate", LoadResString(errStepTextOrFile)
    End If

End Sub

Public Function IncVersionY() As String
    ' The version number for a step is stored in the x.y
    ' format where x is the parent component and y is the
    ' child component of the step. This function will increment
    ' the y component of the step by 1

    On Error GoTo IncVersionYErr

    ' Store the old version number for the step
    Call NewVersion

    mstrVersionNo = Trim$(Str$(GetX(mstrVersionNo))) & gstrVerSeparator & _
            Trim$(Str(GetY(mstrVersionNo) + 1))
    IncVersionY = mstrVersionNo

    Exit Function

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

End Function
Public Function IncVersionX() As String
    ' The version number for a step is stored in the x.y
    ' format where x is the parent component and y is the
    ' child component of the step. This function will increment
    ' the y component of the step by 1 and reset the x component
    ' to 0

    On Error GoTo IncVersionXErr

    ' Store the old version number for the step
    Call NewVersion

    mstrVersionNo = Trim$(Str$(GetX(mstrVersionNo) + 1)) & gstrVerSeparator
& "0"
    IncVersionX = mstrVersionNo

    Exit Function

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

End Function


Private Function GetY(strVersion As String) As Long
    ' The version number for a step is stored in the x.y
    ' format where x is the parent component and y is the
    ' child component of the step. Given an argument of type
```

```
    ' x.y, it returns y

    ' Truncate the fractional part to get the parent component
    ' of the version number (x.y)
    GetY = Val(Mid(strVersion, InStr(strVersion, gstrVerSeparator) + 1))

End Function
Private Function GetX(strVersion As String) As Long
    ' The version number for a step is stored in the x.y
    ' format where x is the parent component and y is the
    ' child component of the step. Given an argument of type
    ' x.y, it returns x

    ' Truncate the fractional part to get the parent component
    ' of the version number (x.y)
    GetX = Val(Left(strVersion, InStr(strVersion, gstrVerSeparator) - 1))

End Function

Public Function Clone(Optional cCloneStep As cStep) As cStep

    ' Creates a copy of a given step

    Dim lngIndex As Long
    Dim cItRec As cIterator
    Dim cItClone As cIterator

    On Error GoTo CloneErr

    If cCloneStep Is Nothing Then
        Set cCloneStep = New cStep
    End If

    ' Copy all the step properties to the newly created step
    ' Initialize the global flag first since subsequent
    ' validations might depend on it
    cCloneStep.GlobalFlag = mblnGlobalFlag
'    cCloneStep.GlobalRunMethod = mintGlobalRunMethod

    cCloneStep.StepType = mintStepType
    cCloneStep.StepId = mlngStepId
    cCloneStep.VersionNo = mstrVersionNo
    cCloneStep.StepLabel = mstrStepLabel
    cCloneStep.StepTextFile = mstrStepTextFile
    cCloneStep.StepText = mstrStepText
    cCloneStep.StartDir = mstrStartDir
    cCloneStep.WorkspaceId = mlngWorkspaceId
    cCloneStep.ParentStepId = mlngParentStepId
    cCloneStep.ParentVersionNo = mstrParentVersionNo
    cCloneStep.StepLevel = mintStepLevel
    cCloneStep.SequenceNo = mintSequenceNo
    cCloneStep.EnabledFlag = mblnEnabledFlag
    cCloneStep.DegreeParallelism = mstrDegreeParallelism
    cCloneStep.ExecutionMechanism = mintExecutionMechanism
    cCloneStep.FailureDetails = mstrFailureDetails
    cCloneStep.ContinuationCriteria = mintContinuationCriteria
    cCloneStep.ArchivedFlag = mblnArchivedFlag
    cCloneStep.OutputFile = mstrOutputFile
'    cCloneStep.LogFile = mstrLogFile
    cCloneStep.ErrorFile = mstrErrorFile
    cCloneStep.IteratorName = mstrIteratorName

    cCloneStep.IndOperation = mintOperation
    cCloneStep.Position = mlngPosition

    Set cCloneStep.NodeDB = mdbsDatabase

    ' Clone all the iterators for the step
    For lngIndex = 0 To mcIterators.Count - 1
        Set cItRec = mcIterators(lngIndex)
        Set cItClone = cItRec.Clone
        cCloneStep.LoadIterator cItClone
    Next lngIndex

    ' And set the return value to the newly created step
    Set Clone = cCloneStep

    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, _
            mstrSource, LoadResString(errCloneFailed)

End Function
'End Sub
'
Public Property Let OutputFile(ByVal vdata As String)

    mstrOutputFile = vdata

End Property

Public Property Get OutputFile() As String

    OutputFile = mstrOutputFile

End Property

'Public Property Let LogFile(ByVal vdata As String)
'
'    mstrLogFile = vdata
'
'End Property
'
'Public Property Get LogFile() As String
'
'    LogFile = mstrLogFile
'
'End Property

Public Property Let ErrorFile(ByVal vdata As String)

    mstrErrorFile = vdata

End Property
Public Property Let IteratorName(ByVal vdata As String)

    mstrIteratorName = vdata

End Property

Public Property Get ErrorFile() As String

    ErrorFile = mstrErrorFile

End Property
Public Property Get IteratorName() As String

    IteratorName = mstrIteratorName

End Property

Public Property Set NodeDB(vdata As Database)

    Set mdbsDatabase = vdata
    Set mcIterators.NodeDB = vdata

End Property
Public Property Get NodeDB() As Database

    Set NodeDB = mdbsDatabase

End Property

Private Function IsStringEmpty(strToCheck As String) As Boolean
```

```
    IsStringEmpty = (strToCheck = gstrEmptyString)

End Function

Public Property Let EnabledFlag(ByVal vdata As Boolean)

    ' The enabled flag must be False for all global steps.
    ' This check must be made by the global step class. Only
    ' generic step validations will be carried out by this
    ' class
    mblnEnabledFlag = vdata

End Property

Public Property Let GlobalFlag(ByVal vdata As Boolean)

    mblnGlobalFlag = vdata

End Property
Public Property Get EnabledFlag() As Boolean

    EnabledFlag = mblnEnabledFlag

End Property

Public Property Let ArchivedFlag(ByVal vdata As Boolean)

    mblnArchivedFlag = vdata

End Property

Public Property Get ArchivedFlag() As Boolean

    ArchivedFlag = mblnArchivedFlag

End Property

Public Property Get GlobalFlag() As Boolean

    GlobalFlag = mblnGlobalFlag

End Property

Public Sub Add()
    ' Inserts a step record into the database - it initializes
    ' the necessary properties for the step and calls InsertStepRec
    ' to do the database work

    On Error GoTo AddErr

    ' A new record would have the deleted_flag turned off!
    mblnArchivedFlag = False

    Call InsertStepRec

    ' If a new version of a step has been created, reset the old version info, since
    ' it's already been saved to the db
    If IsNewVersion() Then
        mbIsNewVersion = False
        msOldVersion = gstrEmptyString
    End If

    Exit Sub

AddErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errAddStepFailed, _
            mstrModuleName & "Add", LoadResString(errAddStepFailed)
End Sub
Private Sub InsertStepRec()
    ' Inserts a step record into the database
    ' It first generates the insert statement using the different
    ' step properties and then executes it

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo InsertStepRecErr

    ' First check if the database object is valid
    Call CheckDB

    ' Check if the step record is valid
    Call Validate

    If IsNewVersion() Then
        Call UpdOldVersionsArchFlg
    End If

    ' Create a temporary querydef object
    strInsert = "insert into att_steps " & _
        "( workspace_id, step_id, version_no, " & _
        " step_label, step_file_name, step_text, start_directory, " & _
        " parent_step_id, parent_version_no, sequence_no, " & _
        " enabled_flag, step_level, " & _
        " degree_parallelism, execution_mechanism, " & _
        " failure_details, " & _
        " continuation_criteria, global_flag, " & _
        " archived_flag, " & _
        " output_file_name, error_file_name, " & _
        " iterator_name ) values ( "

    ' log_file_name,

#If USE_JET Then

    strInsert = strInsert & " [w_id], [s_id], [ver_no], " & _
        " [s_label], [s_file_name], [s_text], [s_start_dir], " & _
        " [p_step_id], [p_version_no], [seq_no], " & _
        " [enabled], [s_level], [deg_parallelism], " & _
        " [exec_mechanism], [fail_dtls], " & _
        " [cont_criteria], [global], [archived], " & _
        " [output_file], [error_file], " & _
        " [it_name] ) "

    ' [log_file],

    Set qy = mdbsDatabase.CreateQueryDef(gstrEmptyString, strInsert)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy)

    qy.Execute dbFailOnError
    qy.Close
#Else

        strInsert = strInsert & Str(mlngWorkspaceId) & ", " & Str(mlngStepId) & _
        ", " & mFieldValue.MakeStringFieldValid(mstrVersionNo)

    ' For fields that may be null, call a function to determine
    ' the string to be appended to the insert statement
    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrStepLabel)
    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrStepTextFile)
    strInsert = strInsert & ", " & mFieldValue.MakeStringFieldValid(mstrStepText)
    strInsert = strInsert & ", " & mFieldValue.MakeStringFieldValid(mstrStartDir)

    strInsert = strInsert & ", " & Str(mlngParentStepId) & _
        ", " & mFieldValue.MakeStringFieldValid(mstrParentVersionNo) & _
        ", " & Str(mintSequenceNo) & _
        ", " & Str(mblnEnabledFlag) & ", " & Str(mintStepLevel)

    strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrDegreeParallelism)
    strInsert = strInsert & ", " & Str(mintExecutionMechanism)
```

```
   strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrFailureDetails) & _
      ", " & Str(mintContinuationCriteria) & _
      ", " & Str(mblnGlobalFlag) & _
      ", " & Str(mblnArchivedFlag)

   strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrOutputFile)
'    strInsert = strInsert & ", " & mFieldValue.MakeStringFieldValid(mstrLogFile)
   strInsert = strInsert & ", " & mFieldValue.MakeStringFieldValid(mstrErrorFile)
   strInsert = strInsert & ", " &
mFieldValue.MakeStringFieldValid(mstrIteratorName)

   strInsert = strInsert & " ) "

   BugMessage strInsert
   mdbsDatabase.Execute strInsert, dbFailOnError

#End If

   Exit Sub

InsertStepRecErr:
   mstrSource = mstrModuleName & "InsertStepRec"
   LogErrors Errors
   On Error GoTo 0
   Err.Raise vbObjectError + errInsertStepFailed, _
         mstrSource, LoadResString(errInsertStepFailed)
End Sub
Private Sub UpdOldVersionsArchFlg()
   ' Updates the archived flag on all old version for the step to True

   Dim sUpdate As String
   Dim qy As DAO.QueryDef

   On Error GoTo UpdOldVersionsArchFlgErr
   mstrSource = mstrModuleName & "UpdOldVersionsArchFlg"

#If USE_JET Then

   sUpdate = "update att_steps " & _
      " set archived_flag = True "

   ' Append the Where clause
   sUpdate = sUpdate & " where step_id = [s_id] " & _
      " and version_no <> [ver_no]"

   Set qy = mdbsDatabase.CreateQueryDef(gstrEmptyString, sUpdate)

   ' Call a procedure to execute the Querydef object
   Call AssignParameters(qy)
   qy.Execute dbFailOnError

   If qy.RecordsAffected = 0 Then
      On Error GoTo 0
      Err.Raise vbObjectError + errModifyStepFailed, _
            mstrSource, LoadResString(errModifyStepFailed)
   End If

   qy.Close

#Else

   sUpdate = "update att_steps " & _
      " set archived_flag = True "

   sUpdate = sUpdate & " where step_id = " & Str(mlngStepId) & _
      " and version_no <> " &
mFieldValue.MakeStringFieldValid(mstrVersionNo)

   BugMessage sUpdate
   mdbsDatabase.Execute sUpdate, dbFailOnError
#End If

   Exit Sub

UpdOldVersionsArchFlgErr:
   mstrSource = mstrModuleName & "UpdOldVersionsArchFlg"
   LogErrors Errors
   On Error GoTo 0
   Err.Raise vbObjectError + errModifyStepFailed, _
         mstrSource, LoadResString(errModifyStepFailed)
End Sub
Public Sub InsertIterator(cItRecord As cIterator)
   ' Inserts the iterator record into the database

   Call cItRecord.Add(mlngStepId, mstrVersionNo)

End Sub
Public Sub UpdateIterator(cItRecord As cIterator)
   ' Updates the iterator record in the database

   Call cItRecord.Update(mlngStepId, mstrVersionNo)

End Sub
Public Sub UpdateIteratorVersion()
   ' Updates the iterator record in the database

   Dim lngIndex As Long
   Dim cTempIt As cIterator

   On Error GoTo UpdateIteratorVersionErr

   For lngIndex = 0 To mcIterators.Count - 1
      ' Increase the array dimension and add the constraint
      ' to it
      Set cTempIt = mcIterators(lngIndex)

      If cTempIt.IndOperation <> DeleteOp Then
         ' Set the operation to indicate an insert
         cTempIt.IndOperation = InsertOp
      End If

   Next lngIndex

   Exit Sub

UpdateIteratorVersionErr:
   mstrSource = mstrModuleName & "UpdateIteratorVersion"
   LogErrors Errors
   On Error GoTo 0
   Err.Raise vbObjectError + errUpdateFailed, _
         mstrSource, LoadResString(errUpdateFailed)

End Sub
Public Sub AddIterator(cItRecord As cIterator)
   ' Adds the iterator record to the collection of iterators
   ' for the step

   Call mcIterators.Add(cItRecord)

End Sub
Public Sub AddAllIterators()
   ' Sets the indicator variable for all iterators to insert

   Dim lngIndex As Long

   For lngIndex = 0 To mcIterators.Count - 1
      mcIterators(lngIndex).Validate
      mcIterators(lngIndex).IndOperation = InsertOp
   Next lngIndex

End Sub

Public Sub LoadIterator(cItRecord As cIterator)
   ' Adds the iterator record to the collection of iterators
   ' for the step
```

```
    Call mcIterators.Load(cItRecord)

End Sub
Public Sub UnloadIterators()
    ' Unloads all iterator records for the step

    Dim lngIndex As Long

    For lngIndex = mcIterators.Count - 1 To 0 Step -1
        ' Calls the collection method to unload the node
        ' from the array
        mcIterators.Unload lngIndex
    Next lngIndex

End Sub
Public Sub ModifyIterator(cItRecord As cIterator)
    ' Modifies the iterator record in the collection

    Call mcIterators.Modify(cItRecord)

End Sub
Public Sub DeleteIterator(cItRecord As cIterator)
    ' Deletes the iterator record from the database

    Call cItRecord.Delete(mlngStepId, mstrVersionNo)

End Sub
Public Sub RemoveIterator(cItRecord As cIterator)
    ' Marks the iterator record in the collection to
    ' indicate a delete

    Call mcIterators.Delete(cItRecord.Position)

End Sub


Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the actual field names. When the parameter names
    ' are the same as the field names, parameters in the
    ' where clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr
    mstrSource = mstrModuleName & "AssignParameters"

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[w_id]"
                prmParam.Value = mlngWorkspaceId
            Case "[s_id]"
                prmParam.Value = mlngStepId
            Case "[ver_no]"
                prmParam.Value = mstrVersionNo
            Case "[s_label]"
                prmParam.Value = mstrStepLabel
            Case "[s_file_name]"
                prmParam.Value = mstrStepTextFile
            Case "[s_text]"
                prmParam.Value = mstrStepText
            Case "[s_start_dir]"
                prmParam.Value = mstrStartDir
            Case "[p_step_id]"
                prmParam.Value = mlngParentStepId
            Case "[p_version_no]"
                prmParam.Value = mstrParentVersionNo
            Case "[seq_no]"
                prmParam.Value = mintSequenceNo
            Case "[enabled]"
                prmParam.Value = mblnEnabledFlag
            Case "[s_level]"
                prmParam.Value = mintStepLevel
            Case "[deg_parallelism]"
                prmParam.Value = mstrDegreeParallelism
            Case "[exec_mechanism]"
                prmParam.Value = mintExecutionMechanism
            Case "[fail_dtls]"
                prmParam.Value = mstrFailureDetails
            Case "[cont_criteria]"
                prmParam.Value = mintContinuationCriteria
            Case "[global]"
                prmParam.Value = mblnGlobalFlag
            Case "[archived]"
                prmParam.Value = mblnArchivedFlag
            Case "[output_file]"
                prmParam.Value = mstrOutputFile
'           Case "[log_file]"
'               prmParam.Value = mstrLogFile
            Case "[error_file]"
                prmParam.Value = mstrErrorFile
            Case "[it_name]"
                prmParam.Value = mstrIteratorName
            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource, _
                    prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrSource, _
                    LoadResString(errInvalidParameter)
        End Select
    Next prmParam

    If qyExec.Parameters("s_id") = 0 Or
StringEmpty(qyExec.Parameters("ver_no")) Then
        WriteError errInvalidParameter, mstrSource
        On Error GoTo 0
        Err.Raise errInvalidParameter, mstrSource,
LoadResString(errInvalidParameter)
    End If

    Exit Sub

AssignParametersErr:

    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrSource, LoadResString(errAssignParametersFailed)

End Sub

Public Sub Modify()

    Dim strUpdate As String
    Dim qy As QueryDef

    On Error GoTo ModifyErr
    mstrSource = mstrModuleName & "Modify"

    ' Check if the database object is valid
    Call CheckDB

    ' Check if the step record is valid
    Call Validate

    ' The step_id and version_no will never be updated -
    ' whenever a step is modified a copy of the old step will
    ' be created with an incremented version_no

#If USE_JET Then

    strUpdate = "update att_steps " & _
        " set step_label = [s_label] " & _
        " , step_file_name = [s_file_name] " & _
```

```
        " , step_text = [s_text] " & _
        " , start_directory = [s_start_dir] " & _
        " , workspace_id = [w_id] " & _
        " , parent_step_id = [p_step_id] " & _
        " , parent_version_no = [p_version_no] " & _
        " , sequence_no = [seq_no] " & _
        " , step_level = [s_level] " & _
        " , enabled_flag = [enabled] " & _
        " , degree_parallelism = [deg_parallelism] " & _
        " , execution_mechanism = [exec_mechanism] " & _
        " , failure_details = [fail_dtls] " & _
        " , continuation_criteria = [cont_criteria] " & _
        " , global_flag = [global] " & _
        " , archived_flag = [archived] " & _
        " , output_file_name = [output_file] " & _
        " , error_file_name = [error_file] " & _
        " , iterator_name = [it_name] "

        ' " , log_file_name = [log_file] " & _

    ' Append the Where clause
    strUpdate = strUpdate & " where step_id = [s_id] " & _
        " and version_no = [ver_no]"

    Set qy = mdbsDatabase.CreateQueryDef(gstrEmptyString, strUpdate)

    ' Call a procedure to execute the Querydef object
    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    If qy.RecordsAffected = 0 Then
        On Error GoTo 0
        Err.Raise vbObjectError + errModifyStepFailed, _
            mstrSource, LoadResString(errModifyStepFailed)
    End If

    qy.Close

#Else

    strUpdate = "update att_steps " & _
        " set step_label = "

    ' For fields that may be null, call a function to determine
    ' the string to be appended to the update statement
    strUpdate = strUpdate & mFieldValue.MakeStringFieldValid(mstrStepLabel)

    strUpdate = strUpdate & " , step_file_name = " &
mFieldValue.MakeStringFieldValid(mstrStepTextFile)
    strUpdate = strUpdate & " , step_text = " &
mFieldValue.MakeStringFieldValid(mstrStepText)
    strUpdate = strUpdate & " , start_directory = " &
mFieldValue.MakeStringFieldValid(mstrStartDir)

    strUpdate = strUpdate & " , workspace_id = " & Str(mlngWorkspaceId) & _
        " , parent_step_id = " & Str(mlngParentStepId) & _
        " , parent_version_no = " &
mFieldValue.MakeStringFieldValid(mstrParentVersionNo) & _
        " , sequence_no = " & Str(mintSequenceNo) & _
        " , step_level = " & Str(mintStepLevel) & _
        " , enabled_flag = " & Str(mblnEnabledFlag) & _
        " , degree_parallelism = " &
mFieldValue.MakeStringFieldValid(mstrDegreeParallelism) & _
        " , execution_mechanism = " & Str(mintExecutionMechanism) & _
        " , failure_details = " &
mFieldValue.MakeStringFieldValid(mstrFailureDetails) & _
        " , continuation_criteria = " & Str(mintContinuationCriteria) & _
        " , global_flag = " & Str(mblnGlobalFlag) & _
        " , archived_flag = " & Str(mblnArchivedFlag) & _
        " , output_file_name = " &
mFieldValue.MakeStringFieldValid(mstrOutputFile) & _
        " , error_file_name = " & mFieldValue.MakeStringFieldValid(mstrErrorFile)
& _
        " , iterator_name = " &
mFieldValue.MakeStringFieldValid(mstrIteratorName)

'       " , log_file_name = " & mFieldValue.MakeStringFieldValid(mstrLogFile) &
_

    strUpdate = strUpdate & " where step_id = " & Str(mlngStepId) & _
        " and version_no = " & mFieldValue.MakeStringFieldValid(mstrVersionNo)

    BugMessage strUpdate
    mdbsDatabase.Execute strUpdate, dbFailOnError
#End If

    Exit Sub

ModifyErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Modify"
    On Error GoTo 0
    Err.Raise vbObjectError + errModifyStepFailed, _
        mstrSource, LoadResString(errModifyStepFailed)
End Sub
Private Sub CheckDB()
    ' Check if the database object has been initialized

    If mdbsDatabase Is Nothing Then
        ShowError errInvalidDB
        On Error GoTo 0
        Err.Raise vbObjectError + errInvalidDB, _
            mstrModuleName, LoadResString(errInvalidDB)
    End If

End Sub

Public Sub Delete()

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteErr

    Call CheckDB

    strDelete = "delete from att_steps " & _
        " where step_id = [s_id] " & _
        " and version_no = [ver_no] "
'   mdbsDatabase.Execute strDelete, dbFailOnError
    Set qy = mdbsDatabase.CreateQueryDef(gstrEmptyString, strDelete)

    Call AssignParameters(qy)
    qy.Execute dbFailOnError

    qy.Close

    Exit Sub

DeleteErr:
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteStepFailed, _
        mstrModuleName & "Delete", LoadResString(errDeleteStepFailed)
End Sub
Public Property Get DegreeParallelism() As String

    DegreeParallelism = mstrDegreeParallelism

End Property
Public Property Get Position() As Long

    Position = mlngPosition

End Property

Public Property Let DegreeParallelism(ByVal vdata As String)
```

```
   ' The degree of parallelism must be zero for all global steps
   ' This check must be made by the global step class. Only
   ' generic step validations will be carried out by this
   ' class
   mstrDegreeParallelism = vdata

End Property

Public Property Let ExecutionMechanism(ByVal vdata As ExecutionMethod)

   BugAssert vdata = gintExecuteODBC Or vdata = gintExecuteShell Or vdata =
gintNoOption, _
         "Execution mechanism invalid"
   mintExecutionMechanism = vdata

End Property

Public Property Let FailureDetails(ByVal vdata As String)

   mstrFailureDetails = vdata

End Property
Public Property Let SequenceNo(ByVal vdata As Integer)
   mintSequenceNo = vdata
End Property

Public Property Let Position(ByVal vdata As Long)
   mlngPosition = vdata
End Property

Public Property Let ParentStepId(ByVal vdata As Long)
   mlngParentStepId = vdata
End Property

Public Property Get SequenceNo() As Integer

   SequenceNo = mintSequenceNo

End Property

Public Property Get StepLevel() As Integer
   StepLevel = mintStepLevel
End Property

Public Property Get ParentVersionNo() As String
   ParentVersionNo = mstrParentVersionNo
End Property

Public Property Let ParentVersionNo(ByVal vdata As String)
   mstrParentVersionNo = vdata
End Property

Public Property Get ParentStepId() As Long
   ParentStepId = mlngParentStepId
End Property

Public Property Let WorkspaceId(ByVal vdata As Long)
   mlngWorkspaceId = vdata
End Property

Public Property Let VersionNo(ByVal vdata As String)
   ' The version number of a step is stored in the x.y format where
   ' x represents a change to the step as a result of modifications
   ' to any of the step properties
   ' y represents a change to the step as a result of modifications
   ' to the sub-steps associated with it. Hence the y-component
   ' of the version will be incremented when a sub-step is added,
   ' modified or deleted
   ' x will be referred to throughout this code as the parent
   ' component of the version and y will be referred to as the
   ' child component of the version
   ' The version information for a step is maintained by the
   ' calling function
   mstrVersionNo = vdata

End Property

Public Property Get StepType() As gintStepType

   On Error GoTo StepTypeErr

   If mintStepType = 0 Then
      ' The step type variable has not been initialized -
      If mblnGlobalFlag Then
         mintStepType = gintGlobalStep
      ElseIf IsStringEmpty(mstrStepText) And _
            IsStringEmpty(mstrStepTextFile) Then
         mintStepType = gintManagerStep
      Else
         mintStepType = gintWorkerStep
      End If
   End If

   StepType = mintStepType

   Exit Property

StepTypeErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "StepType"
   On Error GoTo 0
   Err.Raise vbObjectError + errGetStepTypeFailed, _
         mstrSource, _
         LoadResString(errGetStepTypeFailed)

End Property

Public Property Let StepType(vdata As gintStepType)

   On Error GoTo StepTypeErr

   Select Case vdata
      Case gintGlobalStep, gintManagerStep, gintWorkerStep
         mintStepType = vdata

      Case Else
         On Error GoTo 0
         Err.Raise vbObjectError + errStepTypeInvalid, _
               mstrModuleName & "StepType",
LoadResString(errStepTypeInvalid)
   End Select
   Exit Property

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

End Property

Public Property Get WorkspaceId() As Long
   WorkspaceId = mlngWorkspaceId
End Property

Public Property Get ContinuationCriteria() As ContinuationCriteria

   ContinuationCriteria = mintContinuationCriteria

End Property

Public Property Let ContinuationCriteria(ByVal vdata As ContinuationCriteria)

   ' The Continuation criteria must be null for all global steps
```

```
    ' and non-null for all manager and worker steps
    ' These checks will have to be made by the corresponding
    ' classes - only generic step validations will be made
    ' by this class
    BugAssert vdata = gintOnFailureAbortSiblings Or vdata =
gintOnFailureCompleteSiblings _
            Or vdata = gintOnFailureSkipSiblings Or vdata = gintOnFailureAbort _
            Or vdata = gintOnFailureContinue Or vdata = gintOnFailureAsk _
            Or vdata = gintNoOption, _
            "Invalid continuation criteria"
    mintContinuationCriteria = vdata

End Property
Public Property Get ExecutionMechanism() As ExecutionMethod

    ExecutionMechanism = mintExecutionMechanism

End Property


Public Property Get FailureDetails() As String

    FailureDetails = mstrFailureDetails

End Property

Public Property Let StepText(ByVal vdata As String)
    ' Has to be null for manager steps
    ' The check will have to be made by the user interface or
    ' by the manager step class
    mstrStepText = vdata
End Property
Public Property Let StepLevel(ByVal vdata As Integer)

    ' The step level must be zero for all global steps
    ' This check must be made in the global step class
    mintStepLevel = vdata

End Property
Public Property Get StepText() As String
    StepText = mstrStepText
End Property

Public Property Let StepTextFile(ByVal vdata As String)
    ' Has to be null for manager steps
    ' The check will have to be made by the user interface and
    ' by the manager step class
    mstrStepTextFile = vdata
End Property

Public Property Get StepTextFile() As String
    StepTextFile = mstrStepTextFile
End Property

Public Property Let StepLabel(ByVal vdata As String)
    ' Cannot be null for manager steps
    ' But this check cannot be made here since we do not know
    ' at this point if the step being created is a manager
    ' or a worker step
    ' The check will have to be made by the user interface and
    ' by the manager step class
    mstrStepLabel = vdata
End Property

Public Property Get StepLabel() As String
    StepLabel = mstrStepLabel
End Property

Public Property Let StartDir(ByVal vdata As String)
    mstrStartDir = vdata
End Property

Public Property Get StartDir() As String
    StartDir = mstrStartDir
End Property

Public Property Get VersionNo() As String
    ' The version number of a step is stored in the x.y format where
    ' x represents a change to the step as a result of modifications
    ' to any of the step properties
    ' y represents a change to the step as a result of modifications
    ' to the sub-steps associated with it. Hence the y-component
    ' of the version will be incremented when a sub-step is added,
    ' modified or deleted
    ' x will be referred to throughout this code as the parent
    ' component of the version and y will be referred to as the
    ' child component of the version
    ' The version information for a step is maintained by the
    ' calling function

    VersionNo = mstrVersionNo

End Property

Public Property Get StepId() As Long

    StepId = mlngStepId

End Property
Public Property Get NextStepId() As Long

    Dim lngNextId As Long

    On Error GoTo NextStepIdErr

    ' First check if the database object is valid
    Call CheckDB

    ' Retrieve the next identifier using the sequence class
    Set mStepSeq = New cSequence
    Set mStepSeq.IdDatabase = mdbsDatabase
    mStepSeq.IdentifierColumn = "step_id"
    lngNextId = mStepSeq.Identifier
    Set mStepSeq = Nothing

    NextStepId = lngNextId
    Exit Property

NextStepIdErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "NextStepId"
    On Error GoTo 0
    Err.Raise vbObjectError + errStepIdGetFailed, _
        mstrSource, LoadResString(errStepIdGetFailed)

End Property
Public Property Let StepId(ByVal vdata As Long)

    mlngStepId = vdata

End Property

Private Sub Class_Initialize()

    ' Initialize the operation indicator variable to Query
    ' It will be modified later by the collection class when
    ' inserts, updates or deletes are performed
    mintOperation = QueryOp
    mbIsNewVersion = False
    msOldVersion = gstrEmptyString

    Set mFieldValue = New cStringSM
    Set mcIterators = New cNodeCollections

End Sub

Private Sub Class_Terminate()
```

```
   Set mFieldValue = Nothing
   Set mcIterators = Nothing

End Sub
```

CStepTree.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cStepTree"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cStepTree.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Implements step navigation functions such as determining
'              the child of a step and so on.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cStepTree."
Private mstrSource As String

Public StepRecords As cArrSteps
Public Property Get HasChild(Optional ByVal StepKey As String, _
       Optional ByVal StepId As Long = 0) As Boolean

   Dim lTemp As Long

   HasChild = False
   StepId = GetStepId(StepKey, StepId)

   For lTemp = 0 To StepRecords.StepCount - 1
      If StepRecords(lTemp).StepType <> gintGlobalStep And
StepRecords(lTemp).ParentStepId = StepId Then
         HasChild = True
         Exit For
      End If
   Next lTemp

End Property
Public Property Get ChildStep(Optional ByVal StepKey As String, _
       Optional ByVal StepId As Long = 0) As cStep

   Dim lTemp As Long

   Set ChildStep = Nothing
   StepId = GetStepId(StepKey, StepId)

   For lTemp = 0 To StepRecords.StepCount - 1
      If StepRecords(lTemp).StepType <> gintGlobalStep And
StepRecords(lTemp).ParentStepId = StepId And _
            StepRecords(lTemp).SequenceNo = gintMinSequenceNo Then
         Set ChildStep = StepRecords(lTemp)
         Exit For
      End If
   Next lTemp

End Property
Public Property Get NextStep(Optional ByVal StepKey As String, _
       Optional ByVal StepId As Long = 0) As cStep

   Dim lTemp As Long
   Dim cChildStep As cStep

   Set NextStep = Nothing
   StepId = GetStepId(StepKey, StepId)
   Set cChildStep = StepRecords.QueryStep(StepId)

   For lTemp = 0 To StepRecords.StepCount - 1
      If StepRecords(lTemp).StepType <> gintGlobalStep And _
            StepRecords(lTemp).ParentStepId = cChildStep.ParentStepId And _
            StepRecords(lTemp).SequenceNo = cChildStep.SequenceNo + 1 Then
         Set NextStep = StepRecords(lTemp)
         Exit For
      End If
   Next lTemp

End Property
Private Function GetStepId(Optional ByVal StepKey As String, _
       Optional ByVal StepId As Long = 0) As Long
   If StepId = 0 Then
      If StringEmpty(StepKey) Then
         Err.Raise vbObjectError + errMandatoryParameterMissing, _
               mstrModuleName & "GetStepId",
LoadResString(errMandatoryParameterMissing)
      Else
         GetStepId = IIf(IsLabel(StepKey), 0, MakeIdentifierValid(StepKey))
      End If
   Else
      GetStepId = StepId
   End If
End Function
```

CStringSM.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cStringSM"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cStringSM.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module contains common procedures that can be used
'              to manipulate strings
'              It is called StringSM, since String is a Visual Basic keyword
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cStringSM."

Private mstrText As String

Private Const mstrNullValue = "null"
Private Const mstrSQ = "'"
Private Const mstrEnvVarSeparator = "%"
Public Function InsertEnvVariables( _
       Optional ByVal strComString As String) As String
   ' This function replaces all environment variables in
   ' the passed in string with their values - they are
   ' enclosed by "%"

   Dim intPos As Integer
   Dim intEndPos As Integer
```

```
    Dim strEnvVariable As String
    Dim strValue As String
    Dim strCommand As String

    On Error GoTo InsertEnvVariablesErr
    mstrSource = mstrModuleName & "InsertEnvVariables"

    ' Initialize the return value of the function to the
    ' passed in command
    If IsStringEmpty(strComString) Then
        strCommand = mstrText
    Else
        strCommand = strComString
    End If

    intPos = InStr(strCommand, mstrEnvVarSeparator)
    Do While intPos <> 0
        ' Extract the environment variable from the passed
        ' in string
        intEndPos = InStr(intPos + 1, strCommand, mstrEnvVarSeparator)
        strEnvVariable = Mid(strCommand, intPos + 1, intEndPos - intPos - 1)

        ' Get the value of the variable and call a function
        ' to replace the variable with it's value
        strValue = Environ$(strEnvVariable)
        strCommand = ReplaceSubString(strCommand, _
                mstrEnvVarSeparator & strEnvVariable & mstrEnvVarSeparator, _
                strValue)

        intPos = InStr(strCommand, mstrEnvVarSeparator)
    Loop

    InsertEnvVariables = strCommand
    Exit Function

InsertEnvVariablesErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' Return an empty string
    InsertEnvVariables = gstrEmptyString

End Function
Public Function MakeStringFieldValid( _
        Optional strField As String = gstrEmptyString) As String
    ' Returns a string that can be appended to any insert
    ' or modify (sql) statement
    ' If an argument is not passed to this function, the
    ' default text property is used

    Dim strTemp As String

    On Error GoTo MakeStringFieldValidErr

    If IsStringEmpty(strField) Then
        strTemp = mstrText
    Else
        strTemp = strField
    End If

    ' It checks whether the text is empty
    ' If so, it returns the string, "null"
    If IsStringEmpty(strTemp) Then
        MakeStringFieldValid = mstrNullValue
    Else
        ' Single-quotes have to be replaced by two single-quotes,
        ' since a single-quote is the identifier delimiter
        ' character - call a procedure to do the replace
        strTemp = ReplaceSubString(strTemp, mstrSQ, mstrSQ & mstrSQ)

        ' Replace pipe characters with the corresponding chr function
        strTemp = ReplaceSubString(strTemp, "|", "' & Chr(124) & "')

        ' Enclose the string in single quotes
        MakeStringFieldValid = mstrSQ & strTemp & mstrSQ
    End If

    Exit Function

MakeStringFieldValidErr:
    mstrSource = mstrModuleName & "MakeStringFieldValid"
    LogErrors Errors
    On Error GoTo 0
    Err.Raise vbObjectError + errMakeFieldValidFailed, _
            mstrSource, LoadResString(errMakeFieldValidFailed)

End Function
Public Function MakeDateFieldValid( _
        Optional dtmField As Date = gdtmEmpty) As String
    ' Returns a string that can be appended to any insert
    ' or modify (sql) statement

    ' Enclose the date in single quotes
    MakeDateFieldValid = mstrSQ & dtmField & mstrSQ

End Function

Private Function IsStringEmpty(strToCheck As String) As Boolean

    If strToCheck = gstrEmptyString Then
        IsStringEmpty = True
    Else
        IsStringEmpty = False
    End If

End Function
Public Function ReplaceSubString(ByVal MainString As String, _
        ByVal ReplaceString As String, _
        ByVal ReplaceWith As String) As String

    ' Replaces all occurrences of ReplaceString in MainString with ReplaceWith

    Dim intPos As Integer
    Dim strTemp As String

    On Error GoTo ReplaceSubStringErr

    strTemp = MainString

    intPos = InStr(strTemp, ReplaceString)
    Do While intPos <> 0
        strTemp = Left(strTemp, intPos - 1) & ReplaceWith & _
                Mid(strTemp, intPos + Len(ReplaceString))
        intPos = InStr(intPos + Len(ReplaceString) + 1, strTemp, ReplaceString)
    Loop
    ReplaceSubString = strTemp

    Exit Function

ReplaceSubStringErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "ReplaceSubString"
    On Error GoTo 0
    Err.Raise vbObjectError + errParseStringFailed, _
        mstrSource, _
        LoadResString(errParseStringFailed)

End Function

Public Property Get Text() As String
Attribute Text.VB_UserMemId = 0
    Text = mstrText
End Property

Public Property Let Text(ByVal vdata As String)
    mstrText = vdata
End Property
```

CSubStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cSubStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cSubStep.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     This module encapsulates the properties of sub-steps
'              that are used during the execution of a workspace.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "cSubStep"

Private mlngStepId As Long
Private mintRunning As Integer ' Number of running tasks
Private mintComplete As Integer ' Number of completed tasks
' The last iterator for this sub-step
Private mcLastIterator As cRunItDetails

Public Function NewIteration(cStepRec As cStep) As cIterator
   ' Calls a procedure to determine the next iterator value
   ' for the passed in step - returns the value to be used
   ' in the iteration.
   ' It updates the instance node with the new iteration
   ' for the step.

   Dim cItRec As cIterator

   On Error GoTo NewIterationErr

   ' Call a function that will populate an iterator record
   ' with the iterator values
   Set cItRec = NextIteration(cStepRec)

   ' Initialize the run node with the new iterator
   ' values
   If Not mcLastIterator Is Nothing Then
      If cItRec Is Nothing Then
         mcLastIterator.Value = gstrEmptyString
      Else
         mcLastIterator.Value = cItRec.Value

         ' And if the iterator is a list of values, then update
         ' the sequence number as well
         If mcLastIterator.IteratorType = gintValue Then
            mcLastIterator.Sequence = cItRec.SequenceNo
         End If
      End If
   End If

   Set NewIteration = cItRec
   Set cItRec = Nothing

   Exit Function

NewIterationErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
      LoadResString(errIterateFailed)

End Function
Public Function NextIteration(cStepRec As cStep) As cIterator

   ' Retrieves the next iterator value for the passed in step -
   ' returns an iterator record with the new iterator values

   Dim cItRec As cIterator
   Dim vntIterators As Variant
   Dim lngValue As String

   On Error GoTo NextIterationErr

   vntIterators = cStepRec.Iterators

   If Not mcLastIterator Is Nothing Then
      ' The run node contains the iterator details
      ' Get the next value for the iterator
      If mcLastIterator.IteratorType = gintValue Then
         ' Find the next iterator that appears in the list of
         ' iterator values
         Set cItRec = NextInSequence(vntIterators, mcLastIterator.Sequence)
      Else
         lngValue = CLng(Trim$(mcLastIterator.Value))
         ' Determine whether the new iterator value falls in the
         ' range between From and To
         If (mcLastIterator.RangeStep > 0 And _
               (mcLastIterator.RangeFrom <= mcLastIterator.RangeTo) And _
               (mcLastIterator.RangeStep + lngValue) <= mcLastIterator.RangeTo)
Or _
               (mcLastIterator.RangeStep < 0 And _
               (mcLastIterator.RangeFrom >= mcLastIterator.RangeTo) And _
               (mcLastIterator.RangeStep + lngValue) >= mcLastIterator.RangeTo)
Then
            Set cItRec = New cIterator
            cItRec.Value = Trim$(CStr(mcLastIterator.RangeStep + lngValue))
         Else
            Set cItRec = Nothing
         End If
      End If
   Else
      Set cItRec = Nothing
   End If

   Set NextIteration = cItRec
   Exit Function

NextIterationErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
      LoadResString(errIterateFailed)

End Function
Public Sub InitializeIt(cPendingStep As cStep, _
      ColParameters As cArrParameters, _
      Optional vntIterators As Variant)

   ' Initializes the LastIteration structure with the iterator details for the
   ' passed in step

   On Error GoTo InitializeItErr

   If IsMissing(vntIterators) Then
      vntIterators = cPendingStep.Iterators
   End If

   If IsArray(vntIterators) And Not IsEmpty(vntIterators) Then
      mcLastIterator.IteratorName = cPendingStep.IteratorName
      If vntIterators(LBound(vntIterators)).IteratorType = _
```

```
            gintValue Then
        mcLastIterator.IteratorType = gintValue
        ' Since the sequence numbers begin at 0
        mcLastIterator.Sequence = gintMinIteratorSequence - 1
      Else
        mcLastIterator.IteratorType = gintFrom
        Call InitializeItRange(vntIterators, cPendingStep.WorkspaceId, _
              ColParameters)
      End If
    Else
      Set mcLastIterator = Nothing
    End If

    Exit Sub

InitializeItErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
      LoadResString(errIterateFailed)

End Sub

Private Sub InitializeItRange(vntIterators As Variant, ByVal lWorkspace As
Long, _
      ColParameters As cArrParameters)

    ' Initializes the LastIteration structure for range iterators from the
    ' passed in variant containing the iterator records

    Dim lngIndex As Long
    Dim cItRec As cIterator

    On Error GoTo InitializeItRangeErr

    If IsArray(vntIterators) And Not IsEmpty(vntIterators) Then

      ' Check if the iterator range has been completely initialized
      RangeComplete (vntIterators)

      ' Initialize the Run node with the values for the From,
      ' To and Step boundaries
      For lngIndex = LBound(vntIterators) To UBound(vntIterators)
        Set cItRec = vntIterators(lngIndex)
        Select Case cItRec.IteratorType
          Case gintFrom
            mcLastIterator.RangeFrom = SubstituteParameters(cItRec.Value,
lWorkspace, WspParameters:=ColParameters)
          Case gintTo
            mcLastIterator.RangeTo = SubstituteParameters(cItRec.Value,
lWorkspace, WspParameters:=ColParameters)
          Case gintStep
            mcLastIterator.RangeStep = SubstituteParameters(cItRec.Value,
lWorkspace, WspParameters:=ColParameters)
          Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errTypeInvalid, mstrModuleName, _
                LoadResString(errTypeInvalid)
        End Select
      Next lngIndex

      mcLastIterator.Value = Trim$(CStr(mcLastIterator.RangeFrom -
mcLastIterator.RangeStep))
    End If

    Exit Sub

InitializeItRangeErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
      LoadResString(errIterateFailed)

End Sub
Private Function NextInSequence(vntIterators As Variant, _
      lngOldSequence As Long) As cIterator

    Dim lngIndex As Long
    Dim cItRec As cIterator

    On Error GoTo NextInSequenceErr

    If IsArray(vntIterators) And Not IsEmpty(vntIterators) Then
      For lngIndex = LBound(vntIterators) To UBound(vntIterators)
        Set cItRec = vntIterators(lngIndex)
        If cItRec.IteratorType <> gintValue Then
          On Error GoTo 0
          Err.Raise vbObjectError + errTypeInvalid, mstrModuleName, _
              LoadResString(errTypeInvalid)
        End If
        If cItRec.SequenceNo = lngOldSequence + 1 Then
          Exit For
        End If

      Next lngIndex

      If cItRec.SequenceNo <> lngOldSequence + 1 Then
        Set cItRec = Nothing
      End If
    Else
      Set cItRec = Nothing
    End If

    Set NextInSequence = cItRec

    Exit Function

NextInSequenceErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errIterateFailed, mstrModuleName, _
      LoadResString(errIterateFailed)

End Function

Public Property Get LastIterator() As cRunItDetails

    Set LastIterator = mcLastIterator

End Property
Public Property Set LastIterator(vdata As cRunItDetails)

    Set mcLastIterator = vdata

End Property

Public Property Get TasksRunning() As Integer

    TasksRunning = mintRunning

End Property

Public Property Let TasksRunning(ByVal vdata As Integer)

    mintRunning = vdata

End Property

Public Property Get TasksComplete() As Integer

    TasksComplete = mintComplete

End Property
Public Property Let TasksComplete(ByVal vdata As Integer)
```

```
   mintComplete = vdata

End Property
Public Property Get StepId() As Long

   StepId = mlngStepId

End Property
Public Property Let StepId(ByVal vdata As Long)

   mlngStepId = vdata

End Property
```

CSubSteps.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cSubSteps"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cSubSteps.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module provides a type-safe wrapper around cVector to
'              implement a collection of cSubStep objects.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcSubSteps As cVector

Public Sub Add(ByVal objItem As cSubStep)

   mcSubSteps.Add objItem

End Sub


Public Sub Clear()

   mcSubSteps.Clear

End Sub


Public Function Count() As Long

   Count = mcSubSteps.Count

End Function


Public Function Delete(ByVal lngDelete As Long) As cSubStep

   Set Delete = mcSubSteps.Delete(lngDelete)

End Function


Public Property Get Item(ByVal Position As Long) As cSubStep
Attribute Item.VB_UserMemId = 0

   Set Item = mcSubSteps.Item(Position)

End Property

Private Sub Class_Initialize()

   Set mcSubSteps = New cVector

End Sub

Private Sub Class_Terminate()

   Set mcSubSteps = Nothing

End Sub
```

CTermProcess.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cTermProcess"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cTermProcess.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module raises an event if a completed step exists.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private WithEvents moTimer As cTimerSM
Attribute moTimer.VB_VarHelpID = -1
Private bTermProcessExists As Boolean
Public Event TermProcessExists()

Public Sub ProcessTerminated()

   bTermProcessExists = True
   moTimer.Enabled = True

End Sub

Private Sub Class_Initialize()

   bTermProcessExists = False

   Set moTimer = New cTimerSM
   moTimer.Enabled = False

End Sub

Private Sub Class_Terminate()

   Set moTimer = Nothing

End Sub

Private Sub moTimer_Timer()

   On Error GoTo moTimer_TimerErr

   If bTermProcessExists Then
      RaiseEvent TermProcessExists
   End If

   moTimer.Enabled = False
   bTermProcessExists = False

   Exit Sub
```

```
moTimer_TimerErr:
   LogErrors Errors

End Sub
```

CTermStep.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cTermStep"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cTermStep.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module encapsulates the properties of steps that
'              have completed execution such as status and time of completion.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public TimeComplete As Currency
Public Index As Long
Public InstanceId As Long
Public ExecutionStatus As InstanceStatus
```

CTermSteps.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cTermSteps"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cTermSteps.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module provides a type-safe wrapper around cVector to
'              implement a collection of cTermStep objects. Raises an
'              event if a step that has completed execution exists.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Private mcTermSteps As cVector
Private WithEvents moTimer As cTimerSM
Attribute moTimer.VB_VarHelpID = -1
Public Event TermStepExists(cStepDetails As cTermStep)

Public Sub Add(ByVal citem As cTermStep)

   Call mcTermSteps.Add(citem)
   moTimer.Enabled = True

End Sub

Public Sub Clear()

   mcTermSteps.Clear

End Sub

Public Function Delete()

   Call mcTermSteps.Delete(0)
   ' Disable the timer if there are no more pending events
   If mcTermSteps.Count = 0 Then moTimer.Enabled = False

End Function

Public Property Get Item(ByVal Position As Long) As cTermStep

   Set Item = mcTermSteps(Position)

End Property

Public Function Count() As Long

   Count = mcTermSteps.Count

End Function

Private Sub Class_Initialize()

   Set mcTermSteps = New cVector

   Set moTimer = New cTimerSM
   moTimer.Enabled = False

End Sub

Private Sub Class_Terminate()

   Set mcTermSteps = Nothing
   Set moTimer = Nothing

End Sub

Private Sub moTimer_Timer()

   On Error GoTo moTimer_TimerErr

   If mcTermSteps.Count > 0 Then
      ' Since items are appended to the end of the array
      RaiseEvent TermStepExists(mcTermSteps(0))
   Else
      moTimer.Enabled = False
   End If
   Exit Sub

moTimer_TimerErr:
   LogErrors Errors

End Sub
```

CTimerSM.csl

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cTimerSM"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cTimer.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This module implements a timer.
```

```
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'

Option Explicit

Public Event Timer()

Private Const mnDefaultInterval As Long = 1

Private mnTimerID As Long
Private mnInterval As Long
Private mfEnabled As Boolean

Public Property Get Interval() As Long
   Interval = mnInterval
End Property
Public Property Let Interval(Value As Long)
   If mnInterval <> Value Then
      mnInterval = Value
      If mfEnabled Then
         SetInterval mnInterval, mnTimerID
      End If
   End If
End Property

Public Property Get Enabled() As Boolean
   Enabled = mfEnabled
End Property
Public Property Let Enabled(Value As Boolean)
   If mfEnabled <> Value Then
      If Value Then
         mnTimerID = StartTimer(mnInterval)
         If mnTimerID <> 0 Then
            mfEnabled = True
            'Storing Me in the global would add a reference to Me, which
            '   would prevent Me from being released, which in turn would
            '   prevent my Class_Terminate code from running. To prevent
            '   this, I store a "soft reference" - the collection holds a
            '   pointer to me without incrementing my reference count.
            gcTimerObjects.Add ObjPtr(Me), Str$(mnTimerID)
         End If
      Else
         StopTimer mnTimerID
         mfEnabled = False
         gcTimerObjects.Remove Str$(mnTimerID)
      End If
   End If
End Property

Private Sub Class_Initialize()
   If gcTimerObjects Is Nothing Then Set gcTimerObjects = New Collection
   mnInterval = mnDefaultInterval
End Sub

Private Sub Class_Terminate()
   Enabled = False
End Sub

Friend Sub Tick()
   RaiseEvent Timer
End Sub
```

CVBErrorsSM.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cVBErrorsSM"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
Attribute VB_Ext_KEY = "SavedWithClassBuilder" ,"Yes"
Attribute VB_Ext_KEY = "Top_Level" ,"Yes"
'  FILE:       cVBErrors.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     This module encapsulates the handling of Visual Basic errors.
'              This module does not do any error handling - any error handler
'              will erase the errors object!
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' The Execute class exposes a method, WriteError through which we can write to
the
' error log that is currently being used by the Execute object. Store a reference to
' Execute object locally.
Private mcExecObjRef As EXECUTEDLLLib.Execute
Public Sub WriteError(ByVal ErrorCode As errErrorConstants, _
      Optional ByVal ErrorSource As String = gstrEmptyString, _
      Optional ByVal OptArgs As String = gstrEmptyString)

   Dim sError As String

   sError = "StepMaster Error:" & ErrorCode & vbCrLf &
LoadResString(ErrorCode) & vbCrLf

   If Not StringEmpty(ErrorSource) Then
      sError = sError & "(Source: " & ErrorSource & ")" & vbCrLf
   End If
   sError = sError & OptArgs

   Call LogMessage(sError)

End Sub
Private Function InitErrorString() As String
   ' Initializes a string with all the properties of the
   ' Err object

   Dim strError As String
   Dim errCode As Long

   If Err.Number = 0 Then
      InitErrorString = gstrEmptyString
   Else
   With Err
      If Err.Number > vbObjectError And Err.Number < (vbObjectError + 65536)
Then
         errCode = .Number - vbObjectError
      Else
         errCode = .Number
      End If
      strError = "Error #: " & errCode & vbCrLf
      strError = strError & "Description: " & .Description & vbCrLf
      strError = strError & "Source: " & Err.Source & vbCrLf
   End With

   Debug.Print strError
   InitErrorString = strError
   End If

End Function
Public Sub LogVBErrors()

   Dim strErr As String

   strErr = InitErrorString

   On Error GoTo LogVBErrorsErr

   If Not StringEmpty(strErr) Then
      ' Write an error using the WriteError method of the Execute object.
```

```
      If Not mcExecObjRef Is Nothing Then
         mcExecObjRef.WriteError strErr
      Else
         WriteMessage strErr
      End If
   End If

   Err.Clear

   Exit Sub

LogVBErrorsErr:
   Call LogErrors(Errors)
   ' Since write to the error file for the step has failed, write to the project log
   Call WriteMessage(strErr)

End Sub
Public Sub DisplayErrors()

   Dim strErr As String

   strErr = InitErrorString

   If Not StringEmpty(strErr) Then
      ' Display the error message
      MsgBox strErr
   End If

   Err.Clear

End Sub
Public Sub LogMessage(strMsg As String)

   On Error GoTo LogMessageErr

   ' Write an error using the WriteError method of the Execute object.
   If Not mcExecObjRef Is Nothing Then
      mcExecObjRef.WriteError strMsg
   Else
      WriteMessage strMsg
   End If

   Exit Sub

LogMessageErr:
   Call LogErrors(Errors)
   ' Since write to the error file for the step has failed, write to the project log
   Call WriteMessage(strMsg)

End Sub
Public Property Set ErrorFile(vdata As EXECUTEDLLLib.Execute)

   Set mcExecObjRef = vdata

End Property
Private Sub Class_Terminate()

   Set mcExecObjRef = Nothing

End Sub
```

CVector.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cVector"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cVector.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     This class implements an array of objects.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cVector."

' Array counter
Private mlngCount As Long
Private mcarrItems() As Object

Public Sub Add(ByVal objItem As Object)
   ' Adds the passed in Object variable to the array

   On Error GoTo AddErr

   ReDim Preserve mcarrItems(mlngCount)

   ' Set the newly added element in the array to the
   ' passed in variable
   Set mcarrItems(mlngCount) = objItem
   mlngCount = mlngCount + 1

   Exit Sub

AddErr:
   LogErrors Errors
   gstrSource = mstrModuleName & "Add"
   On Error GoTo 0
   Err.Raise vbObjectError + errLoadInArrayFailed, _
         mstrSource, _
         LoadResString(errLoadInArrayFailed)

End Sub
Public Sub Clear()

   ' Clear the array
   ReDim mcarrItems(0)
   mlngCount = 0

End Sub

Public Function Delete(ByVal lngDelete As Long) As Object

   Dim lngIndex As Long

   On Error GoTo DeleteErr

   If lngDelete < (mlngCount - 1) Then

      ' We want to maintain the order of all items in the
      ' array - so move all remaining elements in the array
      ' up by 1
      For lngIndex = lngDelete To mlngCount - 2
         MoveDown lngIndex
      Next lngIndex

   End If

   ' Return the deleted node
   Set Delete = mcarrItems(mlngCount - 1)

   ' Delete the last Node from the array
   mlngCount = mlngCount - 1
   If mlngCount > 0 Then
      ReDim Preserve mcarrItems(0 To mlngCount - 1)
   Else
```

```
        ReDim mcarrItems(0)
    End If

    Exit Function

DeleteErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteArrayElementFailed, _
            mstrSource, _
            LoadResString(errDeleteArrayElementFailed)

End Function
Public Property Get Item(ByVal Position As Long) As Object
Attribute Item.VB_UserMemId = 0

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mlngCount Then
        Set Item = mcarrItems(Position)
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Set Item(ByVal Position As Long, _
        ByVal Value As Object)

    ' Returns the element at the passed in position in the array
    If Position >= 0 Then
        ' If the passed in position is outside the array
        ' bounds, then resize the array
        If Position >= mlngCount Then
            ReDim Preserve mcarrItems(Position)
            mlngCount = Position + 1
        End If

        ' Set the newly added element in the array to the
        ' passed in variable
        Set mcarrItems(Position) = Value
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Sub MoveUp(ByVal Position As Long)
    ' Moves the element at the passed in position up by 1

    Dim cTemp As Object

    If Position > 0 And Position < mlngCount Then
        Set cTemp = mcarrItems(Position)

        Set mcarrItems(Position) = mcarrItems(Position - 1)
        Set mcarrItems(Position - 1) = cTemp
    End If

End Sub
Public Sub MoveDown(ByVal Position As Long)
    ' Moves the element at the passed in position down by 1

    Dim cTemp As Object

    If Position >= 0 And Position < mlngCount - 1 Then
        Set cTemp = mcarrItems(Position)

        Set mcarrItems(Position) = mcarrItems(Position + 1)
        Set mcarrItems(Position + 1) = cTemp
    End If
End Sub

Public Function Count() As Long

    Count = mlngCount

End Function


Private Sub Class_Initialize()

    mlngCount = 0

End Sub


Private Sub Class_Terminate()

    Call Clear

End Sub
```

CVectorLng.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cVectorLng"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cVectorLng.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    This class implements an array of longs.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cVectorLng."

' Array counter
Private mlngCount As Long
Private mcarrItems() As Long

Public Sub Add(ByVal lngItem As Long)
    ' Adds the passed in long variable to the array

    On Error GoTo AddErr

    ReDim Preserve mcarrItems(mlngCount)

    ' Set the newly added element in the array to the
    ' passed in variable
    mcarrItems(mlngCount) = lngItem
    mlngCount = mlngCount + 1

    Exit Sub

AddErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "Add"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadInArrayFailed, _
            mstrSource, _
```

```
        LoadResString(errLoadInArrayFailed)

End Sub
Public Sub Clear()

   ' Clear the array
   ReDim mcarrItems(0)

End Sub

Public Sub Delete(Optional ByVal Position As Long = -1, _
      Optional ByVal Item As Long = -1)
   ' The user can opt to delete either a specific item in
   ' the list or the item at a specified position. If no
   ' parameters are passed in, we delete the element at
   ' position 0!

   Dim lngDelete As Long
   Dim lngIndex As Long

   On Error GoTo DeleteErr

   If Position = -1 Then
      ' Since we can never store an element at position -1,
      ' we can be sure that the user is trying to delete
      ' a given item
      lngDelete = Find(Item)
   Else
      lngDelete = Position
   End If

   If lngDelete < (mlngCount - 1) Then

      ' We want to maintain the order of all items in the
      ' array - so move all remaining elements in the array
      ' up by 1
      For lngIndex = lngDelete To mlngCount - 2
         MoveDown lngIndex
      Next lngIndex

   End If

   ' Delete the last Node from the array
   mlngCount = mlngCount - 1
   If mlngCount > 0 Then
      ReDim Preserve mcarrItems(0 To mlngCount - 1)
   Else
      ReDim mcarrItems(0)
   End If

   Exit Sub

DeleteErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Delete"
   On Error GoTo 0
   Err.Raise vbObjectError + errDeleteArrayElementFailed, _
         mstrSource, _
         LoadResString(errDeleteArrayElementFailed)

End Sub
Public Function Find(ByVal Item As Long) As Long

   ' Returns the position at which the passed in value occurs
   ' in the array

   Dim lngIndex As Long

   On Error GoTo FindErr

   ' Find the element in the array to be deleted
   For lngIndex = 0 To mlngCount - 1

      If mcarrItems(lngIndex) = Item Then
         Find = lngIndex
         Exit Function
      End If

   Next lngIndex

   Find = -1

   Exit Function

FindErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "Find"
   On Error GoTo 0
   Err.Raise vbObjectError + errItemNotFound, mstrSource, _
      LoadResString(errItemNotFound)

End Function
Public Property Get Item(ByVal Position As Long) As Long
Attribute Item.VB_UserMemId = 0

   ' Returns the element at the passed in position in the array
   If Position >= 0 And Position < mlngCount Then
      Item = mcarrItems(Position)
   Else
      On Error GoTo 0
      Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
         LoadResString(errItemDoesNotExist)
   End If

End Property
Public Property Let Item(ByVal Position As Long, _
      ByVal Value As Long)

   ' Returns the element at the passed in position in the array
   If Position >= 0 Then
      ' If the passed in position is outside the array
      ' bounds, then resize the array
      If Position >= mlngCount Then
         ReDim Preserve mcarrItems(Position)
         mlngCount = Position + 1
      End If

      ' Set the newly added element in the array to the
      ' passed in variable
      mcarrItems(Position) = Value
   Else
      On Error GoTo 0
      Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
         LoadResString(errItemDoesNotExist)
   End If

End Property
Public Sub MoveUp(ByVal Position As Long)
   ' Moves the element at the passed in position up by 1

   Dim lngTemp As Long

   If Position > 0 And Position < mlngCount Then
      lngTemp = mcarrItems(Position)

      mcarrItems(Position) = mcarrItems(Position - 1)
      mcarrItems(Position - 1) = lngTemp
   End If

End Sub
Public Sub MoveDown(ByVal Position As Long)
   ' Moves the element at the passed in position down by 1

   Dim lngTemp As Long

   If Position >= 0 And Position < mlngCount - 1 Then
      lngTemp = mcarrItems(Position)
```

```
        mcarrItems(Position) = mcarrItems(Position + 1)
        mcarrItems(Position + 1) = lngTemp
    End If

End Sub

Public Function Count() As Long

    Count = mlngCount

End Function


Private Sub Class_Initialize()

    mlngCount = 0

End Sub


Private Sub Class_Terminate()

    Call Clear

End Sub
```

CVectorStr.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cVectorStr"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cVectorStr.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:     This class implements an array of strings.
'  Contact:     Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cVectorStr."

' Array counter
Private mlngCount As Long
Private mcarrItems() As String

Public Sub Add(ByVal strItem As String)
    ' Adds the passed in string variable to the array

    On Error GoTo AddErr

    ReDim Preserve mcarrItems(mlngCount)

    ' Set the newly added element in the array to the
    ' passed in variable
    mcarrItems(mlngCount) = strItem
    mlngCount = mlngCount + 1

    Exit Sub

AddErr:
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "Add"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadInArrayFailed, _
          mstrSource, _
          LoadResString(errLoadInArrayFailed)

End Sub
Public Sub Clear()

    ' Clear the array
    ReDim mcarrItems(0)

End Sub

Public Sub Delete(Optional ByVal Position As Long = -1, _
        Optional ByVal Item As String = -1)
    ' The user can opt to delete either a specific item in
    ' the list or the item at a specified position. If no
    ' parameters are passed in, we delete the element at
    ' position 0!

    Dim lngDelete As Long
    Dim lngIndex As Long

    On Error GoTo DeleteErr
    mstrSource = mstrModuleName & "Delete"

    If Position = -1 Then
        ' Since we can never store an element at position -1,
        ' we can be sure that the user is trying to delete
        ' a given item
        lngDelete = Find(Item)
    Else
        lngDelete = Position
    End If

    If lngDelete < (mlngCount - 1) Then

        ' We want to maintain the order of all items in the
        ' array - so move all remaining elements in the array
        ' up by 1
        For lngIndex = lngDelete To mlngCount - 2
            MoveDown lngIndex
        Next lngIndex

    End If

    ' Delete the last Node from the array
    mlngCount = mlngCount - 1
    If mlngCount > 0 Then
        ReDim Preserve mcarrItems(0 To mlngCount - 1)
    Else
        ReDim mcarrItems(0)
    End If

    Exit Sub

DeleteErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "Delete"
    On Error GoTo 0
    Err.Raise vbObjectError + errDeleteArrayElementFailed, _
          mstrSource, _
          LoadResString(errDeleteArrayElementFailed)

End Sub
Public Function Find(ByVal Item As String) As Long

    ' Returns the position at which the passed in value occurs
    ' in the array

    Dim lngIndex As Long

    On Error GoTo FindErr
```

```
    mstrSource = mstrModuleName & "Find"

    ' Find the element in the array to be deleted
    For lngIndex = 0 To mlngCount - 1

        If mcarrItems(lngIndex) = Item Then
            Find = lngIndex
            Exit Function
        End If

    Next lngIndex

    Find = -1

    Exit Function

FindErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "Find"
    On Error GoTo 0
    Err.Raise vbObjectError + errItemNotFound, mstrSource, _
        LoadResString(errItemNotFound)

End Function
Public Property Get Item(ByVal Position As Long) As String
Attribute Item.VB_UserMemId = 0

    ' Returns the element at the passed in position in the array
    If Position >= 0 And Position < mlngCount Then
        Item = mcarrItems(Position)
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Property Let Item(ByVal Position As Long, _
        ByVal Value As String)

    ' Returns the element at the passed in position in the array
    If Position >= 0 Then
        ' If the passed in position is outside the array
        ' bounds, then resize the array
        If Position >= mlngCount Then
            ReDim Preserve mcarrItems(Position)
            mlngCount = Position + 1
        End If

        ' Set the newly added element in the array to the
        ' passed in variable
        mcarrItems(Position) = Value
    Else
        On Error GoTo 0
        Err.Raise vbObjectError + errItemDoesNotExist, mstrSource, _
            LoadResString(errItemDoesNotExist)
    End If

End Property
Public Sub MoveUp(ByVal Position As Long)
    ' Moves the element at the passed in position up by 1

    Dim strTemp As String

    If Position > 0 And Position < mlngCount Then
        strTemp = mcarrItems(Position)

        mcarrItems(Position) = mcarrItems(Position - 1)
        mcarrItems(Position - 1) = strTemp
    End If

End Sub
Public Sub MoveDown(ByVal Position As Long)
    ' Moves the element at the passed in position down by 1

    Dim strTemp As String

    If Position >= 0 And Position < mlngCount - 1 Then
        strTemp = mcarrItems(Position)

        mcarrItems(Position) = mcarrItems(Position + 1)
        mcarrItems(Position + 1) = strTemp
    End If

End Sub

Public Function Count() As Long

    Count = mlngCount

End Function


Private Sub Class_Initialize()

    mlngCount = 0

End Sub
Private Sub Class_Terminate()

    Call Clear

End Sub
```

CWorker.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cWorker"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cWorker.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of a worker step.
'              Implements the cStep class - carries out initializations
'              and validations that are specific to worker steps.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Implements cStep

' Object variable to keep the step reference in
Private mcStep As cStep

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cWorker."
Private Sub cStep_AddAllIterators()

    Call mcStep.AddAllIterators

End Sub

Private Property Let cStep_StartDir(ByVal RHS As String)

    mcStep.StartDir = RHS
```

```
End Property

Private Property Get cStep_StartDir() As String

   cStep_StartDir = mcStep.StartDir

End Property

Private Property Set cStep_NodeDB(RHS As DAO.Database)

   Set mcStep.NodeDB = RHS

End Property

Private Property Get cStep_NodeDB() As DAO.Database

   Set cStep_NodeDB = mcStep.NodeDB

End Property

Private Function cStep_IncVersionY() As String

   cStep_IncVersionY = mcStep.IncVersionY

End Function
Private Function cStep_IsNewVersion() As Boolean
   cStep_IsNewVersion = mcStep.IsNewVersion
End Function
Private Function cStep_OldVersionNo() As String
   cStep_OldVersionNo = mcStep.OldVersionNo
End Function

Private Function cStep_IncVersionX() As String

   cStep_IncVersionX = mcStep.IncVersionX

End Function
Private Sub cStep_UpdateIteratorVersion()

   Call mcStep.UpdateIteratorVersion

End Sub

Private Function cStep_IteratorCount() As Long

   cStep_IteratorCount = mcStep.IteratorCount

End Function

Private Sub cStep_UnloadIterators()

   Call mcStep.UnloadIterators

End Sub

Private Sub cStep_SaveIterators()

   Call mcStep.SaveIterators

End Sub
Private Property Get cStep_IteratorName() As String

   cStep_IteratorName = mcStep.IteratorName

End Property
Private Property Let cStep_IteratorName(ByVal RHS As String)

   mcStep.IteratorName = RHS

End Property

Private Sub cStep_LoadIterator(cItRecord As cIterator)

   Call mcStep.LoadIterator(cItRecord)

End Sub
Private Sub cStep_DeleteIterator(cItRecord As cIterator)

   Call mcStep.DeleteIterator(cItRecord)

End Sub

Private Sub cStep_InsertIterator(cItRecord As cIterator)

   Call mcStep.InsertIterator(cItRecord)

End Sub
Private Function cStep_Iterators() As Variant

   cStep_Iterators = mcStep.Iterators

End Function
Private Sub cStep_ModifyIterator(cItRecord As cIterator)

   Call mcStep.ModifyIterator(cItRecord)

End Sub
Private Sub cStep_RemoveIterator(cItRecord As cIterator)

   Call mcStep.RemoveIterator(cItRecord)

End Sub
Private Sub cStep_UpdateIterator(cItRecord As cIterator)

   Call mcStep.UpdateIterator(cItRecord)

End Sub
Private Sub cStep_AddIterator(cItRecord As cIterator)

   Call mcStep.AddIterator(cItRecord)

End Sub

Private Property Let cStep_Position(ByVal RHS As Long)

   mcStep.Position = RHS

End Property

Private Property Get cStep_Position() As Long

   cStep_Position = mcStep.Position

End Property

Private Function cStep_Clone(Optional cCloneStep As cStep) As cStep

   Dim cNewWorker As cWorker

   Set cNewWorker = New cWorker
   Set cStep_Clone = mcStep.Clone(cNewWorker)

End Function

Private Sub StepTextOrFileEntered()
   ' Checks if either the step text or the name of the file containing
   ' the text has been entered
   ' If both of them are null or both of them are not null,
   ' the worker step is invalid and an error is raised
   If StringEmpty(mcStep.StepText) And StringEmpty(mcStep.StepTextFile)
Then
      ShowError errStepTextAndFileNull
      On Error GoTo 0
      Err.Raise vbObjectError + errStepTextAndFileNull, _
         mstrSource, LoadResString(errStepTextAndFileNull)
   End If

End Sub
```

```
Private Property Get cStep_IndOperation() As Operation

   cStep_IndOperation = mcStep.IndOperation

End Property

Private Property Let cStep_IndOperation(ByVal RHS As Operation)

   mcStep.IndOperation = RHS

End Property

Private Property Get cStep_NextStepId() As Long

   cStep_NextStepId = mcStep.NextStepId

End Property

Private Property Let cStep_OutputFile(ByVal RHS As String)

   mcStep.OutputFile = RHS

End Property

Private Property Get cStep_OutputFile() As String

   cStep_OutputFile = mcStep.OutputFile

End Property

Private Property Let cStep_ErrorFile(ByVal RHS As String)

   mcStep.ErrorFile = RHS

End Property

Private Property Get cStep_ErrorFile() As String

   cStep_ErrorFile = mcStep.ErrorFile

End Property
'Private Property Let cStep_LogFile(ByVal RHS As String)
'
'    mcStep.LogFile = RHS
'
'End Property
'
'Private Property Get cStep_LogFile() As String
'
'    cStep_LogFile = mcStep.LogFile
'
'End Property

Private Property Let cStep_ArchivedFlag(ByVal RHS As Boolean)

   mcStep.ArchivedFlag = RHS

End Property

Private Property Get cStep_ArchivedFlag() As Boolean

   cStep_ArchivedFlag = mcStep.ArchivedFlag

End Property

Private Sub Class_Initialize()

   ' Create the object
   Set mcStep = New cStep

   ' Initialize the object with valid values for a Worker step
   ' The global flag should be the first field to be initialized
   ' since subsequent validations might try to check if the
   ' step being created is global
   mcStep.GlobalFlag = False
'    mcStep.GlobalRunMethod = gintNoOption
   mcStep.StepType = gintWorkerStep

End Sub
Private Sub Class_Terminate()

   ' Remove the step object
   Set mcStep = Nothing

End Sub
Private Sub cStep_Add()

   ' Call a private procedure to see if the step text has been
   ' entered - since a worker step actually executes a step, entry
   ' of the text is mandatory
   Call StepTextOrFileEntered

   ' Call the Add method of the step class to carry out the insert
   mcStep.Add

End Sub

Private Property Get cStep_ContinuationCriteria() As ContinuationCriteria

   cStep_ContinuationCriteria = mcStep.ContinuationCriteria

End Property

Private Property Let cStep_ContinuationCriteria(ByVal RHS As
ContinuationCriteria)

   ' The Continuation criteria must be non-null for all worker steps.
   ' Check if the Continuation Criteria is valid
   Select Case RHS
      Case gintOnFailureAbortSiblings, gintOnFailureCompleteSiblings, _
            gintOnFailureSkipSiblings, gintOnFailureAbort, _
            gintOnFailureContinue, gintOnFailureAsk
         mcStep.ContinuationCriteria = RHS

      Case Else
         On Error GoTo 0
         Err.Raise vbObjectError + errContCriteriaInvalid, _
               mstrModuleName, LoadResString(errContCriteriaInvalid)
   End Select

End Property

Private Property Let cStep_DegreeParallelism(ByVal RHS As String)

   mcStep.DegreeParallelism = RHS

End Property

Private Property Get cStep_DegreeParallelism() As String

   cStep_DegreeParallelism = mcStep.DegreeParallelism

End Property

Private Sub cStep_Delete()

   mcStep.Delete

End Sub

Private Property Get cStep_EnabledFlag() As Boolean

   cStep_EnabledFlag = mcStep.EnabledFlag

End Property

Private Property Let cStep_EnabledFlag(ByVal RHS As Boolean)
```

```
    mcStep.EnabledFlag = RHS

End Property

Private Property Let cStep_ExecutionMechanism(ByVal RHS As
ExecutionMethod)

    On Error GoTo ExecutionMechanismErr
    mstrSource = mstrModuleName & "cStep_ExecutionMechanism"

    Select Case RHS
        Case gintExecuteShell, gintExecuteODBC
            mcStep.ExecutionMechanism = RHS

        Case Else
            On Error GoTo 0
            Err.Raise vbObjectError + errExecutionMechanismInvalid, _
                mstrSource, LoadResString(errExecutionMechanismInvalid)
    End Select

    Exit Property

ExecutionMechanismErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "cStep_ExecutionMechanism"
    On Error GoTo 0
    Err.Raise vbObjectError + errExecutionMechanismLetFailed, _
        mstrSource, LoadResString(errExecutionMechanismLetFailed)

End Property

Private Property Get cStep_ExecutionMechanism() As ExecutionMethod

    cStep_ExecutionMechanism = mcStep.ExecutionMechanism

End Property

Private Property Let cStep_FailureDetails(ByVal RHS As String)

    mcStep.FailureDetails = RHS

End Property

Private Property Get cStep_FailureDetails() As String

    cStep_FailureDetails = mcStep.FailureDetails

End Property

Private Property Get cStep_GlobalFlag() As Boolean

    cStep_GlobalFlag = mcStep.GlobalFlag

End Property

Private Property Let cStep_GlobalFlag(ByVal RHS As Boolean)

    ' Set the global flag to false - this flag is initialized when
    ' an instance of the class is created. Just making sure that
    ' nobody changes the value inadvertently
    mcStep.GlobalFlag = False

End Property
Private Sub cStep_Modify()

    ' Call a private procedure to see if the step text has been
    ' entered - since a worker step actually executes a step, entry
    ' of the text is mandatory
    Call StepTextOrFileEntered

    ' Call the Modify method of the step class to carry out the update
    mcStep.Modify
End Sub

Private Property Let cStep_ParentStepId(ByVal RHS As Long)

    mcStep.ParentStepId = RHS

End Property

Private Property Get cStep_ParentStepId() As Long

    cStep_ParentStepId = mcStep.ParentStepId

End Property

Private Property Let cStep_ParentVersionNo(ByVal RHS As String)

    mcStep.ParentVersionNo = RHS

End Property

Private Property Get cStep_ParentVersionNo() As String

    cStep_ParentVersionNo = mcStep.ParentVersionNo

End Property

Private Property Let cStep_SequenceNo(ByVal RHS As Integer)

    mcStep.SequenceNo = RHS

End Property

Private Property Get cStep_SequenceNo() As Integer

    cStep_SequenceNo = mcStep.SequenceNo

End Property

Private Property Let cStep_StepId(ByVal RHS As Long)

    mcStep.StepId = RHS

End Property

Private Property Get cStep_StepId() As Long

    cStep_StepId = mcStep.StepId

End Property


Private Property Let cStep_StepLabel(ByVal RHS As String)

    mcStep.StepLabel = RHS

End Property

Private Property Get cStep_StepLabel() As String

    cStep_StepLabel = mcStep.StepLabel

End Property


Private Property Let cStep_StepLevel(ByVal RHS As Integer)

    mcStep.StepLevel = RHS

End Property

Private Property Get cStep_StepLevel() As Integer

    cStep_StepLevel = mcStep.StepLevel
```

```
End Property

Private Property Let cStep_StepText(ByVal RHS As String)

   mcStep.StepText = RHS

End Property

Private Property Get cStep_StepText() As String

   cStep_StepText = mcStep.StepText

End Property

Private Property Let cStep_StepTextFile(ByVal RHS As String)

   mcStep.StepTextFile = RHS

End Property

Private Property Get cStep_StepTextFile() As String

   cStep_StepTextFile = mcStep.StepTextFile

End Property

Private Property Let cStep_StepType(RHS As gintStepType)

   mcStep.StepType = gintWorkerStep

End Property

Private Property Get cStep_StepType() As gintStepType

   cStep_StepType = mcStep.StepType

End Property

Private Sub cStep_Validate()
   ' The validate routines for each of the steps will
   ' carry out the specific validations for the type and
   ' call the generic validation routine

   On Error GoTo cStep_ValidateErr

   ' Validations specific to worker steps

   ' Check if the step text or a file name has been
   ' specified
   Call StepTextOrFileEntered

   mcStep.Validate

   Exit Sub

cStep_ValidateErr:
   LogErrors Errors
   mstrSource = mstrModuleName & "cStep_Validate"
   On Error GoTo 0
   Err.Raise vbObjectError + errValidateFailed, _
         mstrSource, _
         LoadResString(errValidateFailed)
End Sub

Private Property Let cStep_VersionNo(ByVal RHS As String)

   mcStep.VersionNo = RHS

End Property

Private Property Get cStep_VersionNo() As String

   cStep_VersionNo = mcStep.VersionNo

End Property

Private Property Let cStep_WorkspaceId(ByVal RHS As Long)

   mcStep.WorkspaceId = RHS

End Property

Private Property Get cStep_WorkspaceId() As Long

   cStep_WorkspaceId = mcStep.WorkspaceId

End Property
```

CWorkspace.cls

```
VERSION 1.0 CLASS
BEGIN
  MultiUse = -1  'True
END
Attribute VB_Name = "cWorkspace"
Attribute VB_GlobalNameSpace = False
Attribute VB_Creatable = True
Attribute VB_PredeclaredId = False
Attribute VB_Exposed = False
'  FILE:       cWorkspace.cls
'              Microsoft TPC-H Kit Ver. 1.00
'              Copyright Microsoft, 1999
'              All Rights Reserved
'
'
'  PURPOSE:    Encapsulates the properties and methods of a workspace.
'              Contains functions to insert, update and delete
'              att_workspaces records from the database.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Local variable(s) to hold property value(s)
Private mlngWorkspaceId As Long
Private mstrWorkspaceName As String
Private mblnArchivedFlag As Boolean
Private mdbsStepMaster As Database

' Used to indicate the source module name when errors
' are raised by this class
Private mstrSource As String
Private Const mstrModuleName As String = "cWorkspace."

' The cSequence class is used to generate unique workspace identifiers
Private mWorkspaceSeq As cSequence

' The StringSM class is used to carry out string operations
Private mFieldValue As cStringSM

Public Function Clone() As cWorkspace

   ' Creates a copy of a given workspace

   Dim cCloneWsp As cWorkspace

   On Error GoTo CloneErr

   Set cCloneWsp = New cWorkspace

   ' Copy all the workspace properties to the newly
   ' created workspace
   cCloneWsp.WorkspaceId = mlngWorkspaceId
   cCloneWsp.WorkspaceName = mstrWorkspaceName
   cCloneWsp.ArchivedFlag = mblnArchivedFlag

   ' And set the return value to the newly created workspace
   Set Clone = cCloneWsp
```

```
    Exit Function

CloneErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "Clone"
    On Error GoTo 0
    Err.Raise vbObjectError + errCloneFailed, _
            mstrSource, LoadResString(errCloneFailed)

End Function

Public Property Let ArchivedFlag(ByVal vdata As Boolean)

    mblnArchivedFlag = vdata

End Property

Public Property Get ArchivedFlag() As Boolean

    ArchivedFlag = mblnArchivedFlag

End Property

Public Property Set WorkDatabase(vdata As Database)

    Set mdbsStepMaster = vdata

End Property

Private Sub WorkspaceNameDuplicate()
    ' Check if the workspace name already exists in the workspace

    Dim rstWorkspace As Recordset
    Dim strSql As String
    Dim qy As DAO.QueryDef

    On Error GoTo WorkspaceNameDuplicateErr
    mstrSource = mstrModuleName & "WorkspaceNameDuplicate"

    ' Create a recordset to retrieve the count of records
    ' having the same workspace name
    strSql = " Select count(*) as workspace_count " & _
        " from att_workspaces " & _
        " where workspace_name = [w_name] " & _
        " and workspace_id <> [w_id] "
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strSql)

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

    Set rstWorkspace = qy.OpenRecordset(dbOpenForwardOnly)

'       mFieldValue.MakeStringFieldValid (mstrWorkspaceName) & _
'       " and workspace_id <> " & _
'       Str(mlngWorkspaceId)
'
'    Set rstWorkspace = mdbsStepMaster.OpenRecordset( _
'       strSQL, dbOpenForwardOnly)

    If rstWorkspace![workspace_count] > 0 Then
        rstWorkspace.Close
        qy.Close
        ShowError errDuplicateWorkspaceName
        On Error GoTo 0
        Err.Raise vbObjectError + errDuplicateWorkspaceName, _
            mstrSource, LoadResString(errDuplicateWorkspaceName)
    End If
    rstWorkspace.Close
    qy.Close

    Exit Sub

WorkspaceNameDuplicateErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "WorkspaceNameDuplicate"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceNameDuplicateFailed, _
        mstrSource, LoadResString(errWorkspaceNameDuplicateFailed)

End Sub
Public Property Let WorkspaceName(vdata As String)

    On Error GoTo WorkspaceNameErr
    mstrSource = mstrModuleName & "WorkspaceName"

    If vdata = gstrEmptyString Then

        On Error GoTo 0
        ' Propogate this error back to the caller
        Err.Raise vbObjectError + errWorkspaceNameMandatory, _
            mstrSource, LoadResString(errWorkspaceNameMandatory)
    Else
        mstrWorkspaceName = vdata
    End If
    Exit Property

WorkspaceNameErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "WorkspaceName"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceNameSetFailed, _
        mstrSource, LoadResString(errWorkspaceNameSetFailed)

End Property

Public Property Let WorkspaceId(vdata As Long)

    On Error GoTo WorkspaceIdErr
    mstrSource = mstrModuleName & "WorkspaceId"

    If (vdata > 0) Then
        mlngWorkspaceId = vdata
    Else
        ' Propogate this error back to the caller
        On Error GoTo 0
        Err.Raise vbObjectError + errWorkspaceIdInvalid, _
            mstrSource, LoadResString(errWorkspaceIdInvalid)
    End If

    Exit Property

WorkspaceIdErr:
    LogErrors Errors
    mstrSource = mstrModuleName & "WorkspaceId"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceIdSetFailed, _
        mstrSource, LoadResString(errWorkspaceIdSetFailed)

End Property

Public Sub AddWorkspace()

    Dim strInsert As String
    Dim qy As DAO.QueryDef

    On Error GoTo AddWorkspaceErr

    ' Retrieve the next identifier using the sequence class
    Set mWorkspaceSeq = New cSequence
    Set mWorkspaceSeq.IdDatabase = mdbsStepMaster
    mWorkspaceSeq.IdentifierColumn = FLD_ID_WORKSPACE
    mlngWorkspaceId = mWorkspaceSeq.Identifier
    Set mWorkspaceSeq = Nothing

    ' Call procedure to raise an error if the Workspace name
    ' already exists in the db
    Call WorkspaceNameDuplicate
```

```
    ' A new record will have the archived_flag turned off
    mblnArchivedFlag = False

    ' Create a temporary querydef object
    strInsert = "insert into att_workspaces " & _
            "( workspace_id, workspace_name, " & _
            " archived_flag ) " & _
            " values ( [w_id], [w_name], [archived] ) "
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strInsert)

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

    qy.Execute dbFailOnError
    qy.Close

'    strInsert = "insert into att_workspaces " & _
'            "( workspace_id, workspace_name, " & _
'            " archived_flag ) " & _
'            " values ( " & _
'            Str(mlngWorkspaceId) & _
'            ", " & mFieldValue.MakeStringFieldValid(mstrWorkspaceName) & _
'            ", " & Str(mblnArchivedFlag) & _
'            " ) "
'    mdbsStepMaster.Execute strInsert, dbFailOnError
'
    Exit Sub

AddWorkspaceErr:

    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "AddWorkspace"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceInsertFailed, _
        mstrSource, LoadResString(errWorkspaceInsertFailed)

End Sub
Private Sub AssignParameters(qyExec As DAO.QueryDef)
    ' Assigns values to the parameters in the querydef object
    ' The parameter names are cryptic to make them different
    ' from the field names. When the parameter names are
    ' the same as the field names, parameters in the where
    ' clause do not get created.

    Dim prmParam As DAO.Parameter

    On Error GoTo AssignParametersErr
    mstrSource = mstrModuleName & "AssignParameters"

    For Each prmParam In qyExec.Parameters
        Select Case prmParam.Name
            Case "[w_id]"
                prmParam.Value = mlngWorkspaceId

            Case "[w_name]"
                prmParam.Value = mstrWorkspaceName

            Case "[archived]"
                prmParam.Value = mblnArchivedFlag

            Case Else
                ' Write the parameter name that is faulty
                WriteError errInvalidParameter, mstrSource, _
                        prmParam.Name
                On Error GoTo 0
                Err.Raise errInvalidParameter, mstrSource, _
                        LoadResString(errInvalidParameter)
        End Select
    Next prmParam

    Exit Sub

AssignParametersErr:
    mstrSource = mstrModuleName & "AssignParameters"
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errAssignParametersFailed, _
        mstrSource, LoadResString(errAssignParametersFailed)

End Sub
Public Sub DeleteWorkspace()

    Dim strDelete As String
    Dim qy As DAO.QueryDef

    On Error GoTo DeleteWorkspaceErr

    strDelete = "delete from att_workspaces " & _
        " where workspace_id = [w_id]"
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strDelete)

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

    qy.Execute dbFailOnError
    qy.Close

'    mdbsStepMaster.Execute strDelete, dbFailOnError
'        " where workspace_id = " & _
'        Str(mlngWorkspaceId)

    Exit Sub

DeleteWorkspaceErr:
    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "DeleteWorkspace"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceDeleteFailed, _
        mstrSource, LoadResString(errWorkspaceDeleteFailed)
End Sub

Public Sub ModifyWorkspace()

    Dim strUpdate As String
    Dim qy As DAO.QueryDef

    On Error GoTo ModifyWorkspaceErr

    ' Call procedure to raise an error if the Workspace name
    ' already exists in the db
    Call WorkspaceNameDuplicate

    strUpdate = "update att_workspaces " & _
        " set workspace_name = [w_name] " & _
        ", archived_flag = [archived] " & _
        " where workspace_id = [w_id] "
    Set qy = mdbsStepMaster.CreateQueryDef(gstrEmptyString, strUpdate)

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

    qy.Execute dbFailOnError
    qy.Close

'    strUpdate = "update att_workspaces " & _
'        " set workspace_name = " & _
'        mFieldValue.MakeStringFieldValid(mstrWorkspaceName) & _
'        ", archived_flag = " & _
'        Str(mblnArchivedFlag) & _
'        " where workspace_id = " & _
'        Str(mlngWorkspaceId)
'
'    mdbsStepMaster.Execute strUpdate, dbFailOnError
'
    Exit Sub
```

```
ModifyWorkspaceErr:

    Call LogErrors(Errors)
    mstrSource = mstrModuleName & "ModifyWorkspace"
    On Error GoTo 0
    Err.Raise vbObjectError + errWorkspaceUpdateFailed, _
        mstrSource, LoadResString(errWorkspaceUpdateFailed)

End Sub
Public Property Get WorkspaceName() As String

    WorkspaceName = mstrWorkspaceName

End Property

Public Property Get WorkspaceId() As Long

    WorkspaceId = mlngWorkspaceId

End Property

Private Sub Class_Initialize()

    ' Each function will append it's own name to this
    ' variable
    mstrSource = "cWorkspace."

    Set mFieldValue = New cStringSM

End Sub

Private Sub Class_Terminate()

    Set mdbsStepMaster = Nothing
    Set mFieldValue = Nothing

End Sub
```

DatabaseSM.bas

```
Attribute VB_Name = "DatabaseSM"
'  FILE:      DatabaseSM.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Contains all the database initialization/cleanup
'             procedures for the project. Also contains upgrade
'             database upgrade functions.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
' This module is called DatabaseSM, since Database is a standard
' Visual Basic object and we want to avoid any confusion with it.

Option Explicit

Public wrkJet As Workspace
Public dbsAttTool As Database
Public gblnDbOpen As Boolean
Public gRunEngine As rdoEngine

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "DatabaseSM."
Public Const gsDefDBFileExt As String = ".stp"
Private Const msDefDBFile As String = "\SMData" & gsDefDBFileExt

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

Private Const STEPMASTER_SETTINGS_VAL_NAME_DBFILE As String =
"WorkspaceFile"

Public Const DEF_NO_COUNT_DISPLAY          As Boolean = False
Public Const DEF_NO_EXECUTE                As Boolean = False
Public Const DEF_PARSE_QUERY_ONLY          As Boolean = False
Public Const DEF_ANSI_QUOTED_IDENTIFIERS   As Boolean = False
Public Const DEF_ANSI_NULLS                As Boolean = True
Public Const DEF_SHOW_QUERY_PLAN           As Boolean = False
Public Const DEF_SHOW_STATS_TIME           As Boolean = False
Public Const DEF_SHOW_STATS_IO             As Boolean = False
Public Const DEF_PARSE_ODBC_MSG_PREFIXES   As Boolean = True
Public Const DEF_ROW_COUNT                 As Long = 0
Public Const DEF_TSQL_BATCH_SEPARATOR      As String = "GO"
Public Const DEF_QUERY_TIME_OUT            As Long = 0
Public Const DEF_SERVER_LANGUAGE           As String = "(Default)"
Public Const DEF_CHARACTER_TRANSLATION     As Boolean = True
Public Const DEF_REGIONAL_SETTINGS         As Boolean = False

Public Const PARAM_DEFAULT_DIR             As String = "DEFAULT_DIR"
Public Const PARAM_DEFAULT_DIR_DESC        As String = "Default
destination directory " & _
            "for all output and error files. If it is blank, the StepMaster installation
directory will be used."

Public Const PARAM_RUN_ID                  As String = "RUN_ID"
Public Const PARAM_RUN_ID_DESC             As String = "The run identifier
for a run. " & _
            "Any modifications will be overwritten before each run."

Public Const PARAM_OUTPUT_DIR              As String = "OUTPUT_DIR"
Public Const PARAM_OUTPUT_DIR_DESC         As String = "The output
directory for a run. " & _
            "Any modifications will be overwritten before each run."

Public Const CONNECTION_STRINGS_TO_NAME_SUFFIX As String =
"_NAME"

Private Const TBL_RUN_STEP_HDR As String = "run_header"
Private Const TBL_RUN_STEP_DTLS As String = "run_step_details"
Public Const TBL_CONNECTION_DTLS As String = "connection_dtls"
Public Const TBL_CONNECTION_STRINGS As String =
"workspace_connections"
Public Const TBL_STEPS As String = "att_steps"

Public Const FLD_ID_CONN_NAME As String = "connection_name_id"
Public Const FLD_ID_WORKSPACE As String = "workspace_id"
Public Const FLD_ID_STEP As String = "step_id"
Public Const FLD_ID_PARAMETER As String = "parameter_id"

Public Const FLD_CONN_DTL_CONNECTION_NAME As String =
"connection_name"
Public Const FLD_CONN_DTL_CONNECTION_STRING As String =
"connection_string_name"
Public Const FLD_CONN_DTL_CONNECTION_TYPE As String =
"connection_type"

Public Const FLD_CONN_STR_CONNECTION_NAME As String =
"connection_name"

Public Const FLD_STEPS_EXEC_MECHANISM As String =
"execution_mechanism"
Public Const FLD_STEPS_EXEC_DTL As String = "start_directory"
Public Const FLD_STEPS_VERSION_NO As String = "version_no"

Public Const DATA_TYPE_CURRENCY As String = "CURRENCY"
Public Const DATA_TYPE_LONG As String = "Long"
Public Const DATA_TYPE_INTEGER As String = "INTEGER"
Public Const DATA_TYPE_TEXT255 As String = "Text(255)"

Private Sub InsertBuiltInParameter(dbFile As Database, sParamName As String, _
    sParamValue As String, sParamDesc As String)

    Dim sBuf As String
    Dim cTempStr As New cStringSM
    Dim lId As Long
```

```
   Dim rTemp As DAO.Recordset
   Dim rParam As DAO.Recordset
   Dim cTempSeq As cSequence

   ' Create the passed in built-in parameter, for each workspace in the db
   Set cTempSeq = New cSequence
   Set cTempSeq.IdDatabase = dbFile
   cTempSeq.IdentifierColumn = FLD_ID_PARAMETER

   sBuf = "select * from att_workspaces "
   Set rTemp = dbFile.OpenRecordset(sBuf, dbOpenSnapshot)
   If rTemp.RecordCount <> 0 Then
      rTemp.MoveFirst

      While Not rTemp.EOF
         sBuf = "select * from workspace_parameters " & _
               " where workspace_id = " & Str(rTemp!workspace_id) & _
               " and parameter_name = " &
cTempStr.MakeStringFieldValid(sParamName)
         Set rParam = dbFile.OpenRecordset(sBuf, dbOpenSnapshot)
         If rParam.RecordCount <> 0 Then
            rParam.MoveFirst
            ' Since the parameter already exists, change it to a built-in type
            sBuf = "update workspace_parameters " & _
                  " set parameter_type = " & CStr(gintParameterBuiltIn) & _
                  ", description = " & cTempStr.MakeStringFieldValid(sParamDesc)
& _
                  " where workspace_id = " & Str(rTemp!workspace_id) & _
                  " and parameter_id = " & Str(rParam!parameter_id)
         Else
            ' Else, insert a parameter record
            lId = cTempSeq.Identifier
            sBuf = "insert into workspace_parameters " & _
                  "( workspace_id, parameter_id, " & _
                  " parameter_name, parameter_value, " & _
                  " description, parameter_type ) " & _
                  " values ( " & _
                  Str(rTemp!workspace_id) & ", " & Str(lId) & ", " & _
                  cTempStr.MakeStringFieldValid(sParamName) & ", " & _
                  cTempStr.MakeStringFieldValid(sParamValue) & ", " & _
                  cTempStr.MakeStringFieldValid(sParamDesc) & ", " & _
                  CStr(gintParameterBuiltIn) & _
                  " ) "
         End If
         dbFile.Execute sBuf, dbFailOnError
         rParam.Close

         rTemp.MoveNext
      Wend
   End If
   rTemp.Close

End Sub
Public Sub InitRunEngine()

   Set gRunEngine = New rdoEngine
   gRunEngine.rdoDefaultCursorDriver = rdUseServer

End Sub

Public Function DefaultDBFile() As String
   DefaultDBFile = GetSetting(App.Title, "Settings",
STEPMASTER_SETTINGS_VAL_NAME_DBFILE, App.Path & msDefDBFile)
End Function

Public Sub CloseDatabase()

   Dim dbsInstance As Database
   Dim recInstance As Recordset

   On Error GoTo CloseDatabaseErr

   ' Close all open recordsets and databases in the workspace
   For Each dbsInstance In wrkJet.Databases
      For Each recInstance In dbsAttTool.Recordsets
         recInstance.Close
      Next recInstance
      dbsInstance.Close

   Next dbsInstance

   Set dbsAttTool = Nothing

   gblnDbOpen = False
   wrkJet.Close

   Exit Sub

CloseDatabaseErr:

   Call LogErrors(Errors)
   Resume Next

End Sub

Private Function NoDbChanges(sVerTo As String, sVerFrom As String) As
Boolean

   If sVerTo = gsVersion242 And sVerFrom = gsVersion241 Then
      NoDbChanges = True
   ElseIf sVerTo = gsVersion242 And sVerFrom = gsVersion24 Then
      NoDbChanges = True
   ElseIf sVerTo = gsVersion253 And sVerFrom = gsVersion251 Then
      NoDbChanges = True
   ElseIf sVerTo = gsVersion255 And sVerFrom = gsVersion251 Then
      NoDbChanges = True
   Else
      NoDbChanges = False
   End If

End Function

Public Function SMOpenDatabase(Optional strDbName As String =
gstrEmptyString) As Boolean
   Dim sVersion As String
   Dim bOpeningDb As Boolean ' This flag is used to check if OpenDatabase
failed

   On Error GoTo OpenDatabaseErr

   bOpeningDb = False
   SMOpenDatabase = False

   ' Create Microsoft Jet Workspace object.
   If Not gblnDbOpen Then
      Set wrkJet = CreateWorkspace("att_tool_workspace_setup", "admin",
gstrEmptyString, dbUseJet)
   End If

   ' Prompt the user for the database file if it is not passed in
   If StringEmpty(strDbName) Then
      strDbName = BrowseDBFile
      If StringEmpty(strDbName) Then
         Exit Function
      End If
   End If

   Do
      If gblnDbOpen Then
#If Not RUN_ONLY Then
         CloseOpenWorkspaces
#End If
         Set wrkJet = CreateWorkspace("att_tool_workspace_setup", "admin",
gstrEmptyString, dbUseJet)
      End If

      ' Toggle the bOpeningDb flag around the OpenDatabase method - the value
```

```
        ' of this flag will be checked by the error handler to determine if it is
        ' the OpenDatabase that failed.
        BugMessage "DB File: " & strDbName

        bOpeningDb = True
        ' Open the database for exclusive use
        Set dbsAttTool = wrkJet.OpenDatabase(strDbName, Options:=True)
        bOpeningDb = False

        If dbsAttTool Is Nothing Then
            ' If the file is not present in the directory, display
            ' an error and ask the user to enter a new path
            Call ShowError(errOpenDbFailed, OptArgs:=strDbName)

            strDbName = BrowseDBFile
        Else
            sVersion = DBVersion(dbsAttTool)

            ' Make sure the application and db version numbers match
            If sVersion = gsVersion Then
                Call InitializeData(strDbName)
                gblnDbOpen = True
                SMOpenDatabase = True
            Else
                If UpgradeDb(wrkJet, dbsAttTool, gsVersion, sVersion) Then
                    Call InitializeData(strDbName)
                    gblnDbOpen = True
                    SMOpenDatabase = True
                Else
                    dbsAttTool.Close
                    Set dbsAttTool = Nothing

                    ShowError errVersionMismatch, _
                        OptArgs:=" Please install Version '" & gsVersion & "' of the
workspace definition file."
                    strDbName = BrowseDBFile
                End If
            End If
        End If
    Loop While gblnDbOpen = False And Not StringEmpty(strDbName)

    Exit Function

OpenDatabaseErr:
    Call DisplayErrors(Errors)

    ' If the OpenDatabase failed, continue
    If bOpeningDb Then
        Resume Next
    End If

    Call ShowError(errOpenDbFailed, OptArgs:=strDbName)

End Function
Private Sub InitializeData(sDb As String)

    Set gcParameters = New cArrParameters
    Set gcParameters.ParamDatabase = dbsAttTool

    Set gcSteps = New cArrSteps
    Set gcSteps.StepDB = dbsAttTool

    Set gcConstraints = New cArrConstraints
    Set gcConstraints.ConstraintDB = dbsAttTool

    Set gcConnections = New cConnections
    Set gcConnections.ConnDb = dbsAttTool

    Set gcConnDtls = New cConnDtls
    Set gcConnDtls.ConnDb = dbsAttTool

    ' Disable the error handler since this is not a critical step
    On Error GoTo 0
    SaveSetting App.Title, "Settings",
STEPMASTER_SETTINGS_VAL_NAME_DBFILE, sDb
End Sub
Private Sub UpdateContinuationCriteria(dbFile As DAO.Database)

    Dim qyTemp As DAO.QueryDef
    Dim sBuf As String

    On Error GoTo UpdateContinuationCriteriaErr

    sBuf = "Since this version of the executable incorporates failure processing, "
& _
            "the upgrade will update the On Failure field for each of the steps " & _
            "to 'Continue' to be compatible with the existing behaviour. " & _
            "Proceed?"
    If Not Confirm(Buttons:=vbYesNo, strMessage:=sBuf, strTitle:="Upgrade
database") Then
        Exit Sub
    End If

    ' Create a recordset object to retrieve all steps for
    ' the given workspace
    sBuf = " update att_steps a " & _
        " set continuation_criteria = " & CStr(gintOnFailureContinue) & _
        " where archived_flag = [archived] "

    ' Find the highest X-component of the version number
    sBuf = sBuf & " AND cint( mid( version_no, 1, instr( version_no, " & gstrDQ
& gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS d " & _
        " WHERE a.step_id = d.step_id ) "

    ' Find the highest Y-component of the version number for the highest
X-component
    sBuf = sBuf & " AND cint( mid( version_no, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) + 1 ) ) = " & _
        " ( select max( cint( mid( version_no, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) + 1 ) ) ) " & _
        " from att_steps AS b " & _
        " Where a.step_id = b.step_id " & _
        " AND cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS c " & _
        " WHERE a.step_id = c.step_id ) ) "

    ' Create a temporary Querydef object
    Set qyTemp = dbFile.CreateQueryDef(gstrEmptyString, sBuf)
    qyTemp.Parameters("archived").Value = False

    qyTemp.Execute dbFailOnError
    qyTemp.Close

    Exit Sub

UpdateContinuationCriteriaErr:
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errModifyStepFailed, mstrModuleName, _
            LoadResString(errModifyStepFailed)

End Sub

Private Sub UpdateDbDtls(dbFile As Database, sNewVersion As String)

    Dim sSql As String
    Dim cTemp As New cStringSM

    On Error GoTo UpdateDbDtlsErr

    sSql = "update db_details " & _
            " set db_version = " & cTemp.MakeStringFieldValid(sNewVersion)
```

```
    dbFile.Execute sSql, dbFailOnError

    Exit Sub

UpdateDbDtlsErr:
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
        LoadResString(errUpgradeFailed)

End Sub

Private Sub Upgrade10to21(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sSql As String

    On Error GoTo Upgrade10to21Err

    Call UpdateDbDtls(dbFile, sVersion)

    Call UpdateContinuationCriteria(dbFile)

    Exit Sub

Upgrade10to21Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
        LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade21to23(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sBuf As String
    Dim cTempStr As New cStringSM

    On Error GoTo Upgrade21to23Err

    ' Add a parameter type field and a description field to the parameter table
    sBuf = "alter table workspace_parameters " & _
        " add column description TEXT(255) "
    dbFile.Execute sBuf, dbFailOnError

    sBuf = "alter table workspace_parameters " & _
        " add column parameter_type INTEGER "
    dbFile.Execute sBuf, dbFailOnError

    ' Initialize the parameter type on all parameters to indicate generic parameters
    sBuf = "update workspace_parameters " & _
        " set parameter_type = " & CStr(gintParameterGeneric)
    dbFile.Execute sBuf, dbFailOnError

    sBuf = "Release 2.3 onwards, connection string parameters will be " & _
        "displayed in a separate node. After this upgrade, all connection " & _
        "string parameters will appear under the Globals/Connection Strings " & _
        "node in the workspace. "
    Call MsgBox(sBuf, vbOKOnly + vbApplicationModal, "Upgrade database")

    ' Update the parameter type on all parameters that look like db connection
strings
    sBuf = "update workspace_parameters " & _
        " set parameter_type = " & CStr(gintParameterConnect) & _
        " where UCase(parameter_value) like '*DRIVER*' " & _
        " or UCase(parameter_value) like '*DSN*'"
    dbFile.Execute sBuf, dbFailOnError

    ' Add an elapsed time field to the run_step_details table - this field is
    ' needed to store the elapsed time in milliseconds.
    sBuf = "alter table run_step_details " & _
        " add column elapsed_time LONG "
    dbFile.Execute sBuf, dbFailOnError

    ' The failure_details field has some data for the case when an ODBC failure
    ' threshold was specified. Since that's no longer relevant, update the
failure_details
    ' field for records with failure_criteria = gintFailureODBC to empty.
    ' failure_criteria = gintFailureODBC = 1
    sBuf = "update att_steps " & _
        " set failure_details = " &
cTempStr.MakeStringFieldValid(gstrEmptyString) & _
        " where failure_criteria = '1'"
    dbFile.Execute sBuf, dbFailOnError

    Call UpdateDbDtls(dbFile, sVersion)

    UpgradeWsp.CommitTrans

    On Error GoTo DropColumnErr

    UpgradeWsp.BeginTrans

    ' This ddl cannot be in the same transaction as the failure_details update
    ' But we can do this in a separate transaction since we do not expect this
    ' statement to fail - AND, it doesn't matter if this transaction fails
    ' Drop the failure_criteria column from the att_steps table
    sBuf = "alter table att_steps " & _
        " drop column failure_criteria "
    dbFile.Execute sBuf, dbFailOnError

    Exit Sub

DropColumnErr:
    Call LogErrors(Errors)
    ShowError errDeleteColumnFailed
    Exit Sub

Upgrade21to23Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
        LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade23to24(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sBuf As String
    Dim lId As Long
    Dim rTemp As DAO.Recordset
    Dim cTempStr As New cStringSM

    On Error GoTo Upgrade23to24Err

    ' Add a new table for connection properties
    sBuf = CreateConnectionsTableScript()
    ' TODO: Not sure of column sizes for row count, tsql_batch_separator and
server_language
    dbFile.Execute sBuf, dbFailOnError

    ' Move all connection parameters from the parameter table to the connections
tables
    ' Insert default values for the newly added connection properties
    sBuf = "select * from workspace_parameters " & _
        "where parameter_type = " & CStr(gintParameterConnect)
    Set rTemp = dbFile.OpenRecordset(sBuf, dbOpenSnapshot)
    lId = 1
    If rTemp.RecordCount <> 0 Then
        rTemp.MoveFirst

        While Not rTemp.EOF
            sBuf = "insert into workspace_connections " & _
            "( workspace_id, connection_id, " & _
            "connection_name, connection_value, " & _
            "description, no_count_display, " & _
            "no_execute, parse_query_only, " & _
            "ANSI_quoted_identifiers, ANSI_nulls, " & _
```

```
            "show_query_plan, show_stats_time, " & _
            "show_stats_io, parse_odbc_msg_prefixes, " & _
            "row_count, tsql_batch_separator, " & _
            "query_time_out, server_language, " & _
            "character_translation, regional_settings ) " & _
           " values ( " & _
           Str(rTemp!workspace_id) & ", " & Str(lId) & ", " & _
           cTempStr.MakeStringFieldValid("" & rTemp!parameter_name) & ", " & _
           cTempStr.MakeStringFieldValid("" & rTemp!parameter_value) & ", " & _
           cTempStr.MakeStringFieldValid("" & rTemp!Description) & ", " & _
           Str(DEF_NO_COUNT_DISPLAY) & ", " & _
           Str(DEF_NO_EXECUTE) & ", " & Str(DEF_PARSE_QUERY_ONLY)
& ", " & _
           Str(DEF_ANSI_QUOTED_IDENTIFIERS) & ", " &
Str(DEF_ANSI_NULLS) & ", " & _
           Str(DEF_SHOW_QUERY_PLAN) & ", " &
Str(DEF_SHOW_STATS_TIME) & ", " & _
           Str(DEF_SHOW_STATS_IO) & ", " &
Str(DEF_PARSE_ODBC_MSG_PREFIXES) & ", " & _
           Str(DEF_ROW_COUNT) & ", " &
cTempStr.MakeStringFieldValid(DEF_TSQL_BATCH_SEPARATOR) & ", " &
_
           Str(DEF_QUERY_TIME_OUT) & ", " &
cTempStr.MakeStringFieldValid(DEF_SERVER_LANGUAGE) & ", " & _
           Str(DEF_CHARACTER_TRANSLATION) & ", " &
Str(DEF_REGIONAL_SETTINGS) & _
           " ) "
           dbFile.Execute sBuf, dbFailOnError

           lId = lId + 1
           rTemp.MoveNext
        Wend
    End If
    rTemp.Close

    ' Add an identifier column for the connection_id field
    sBuf = "alter table att_identifiers " & _
           " add column connection_id long "
    dbFile.Execute sBuf, dbFailOnError

    ' Initialize the value of the connection identifier, initialized above
    sBuf = "update att_identifiers " & _
           " set connection_id = " & Str(lId)
    dbFile.Execute sBuf, dbFailOnError

    ' Delete all connection strings from the parameter table
    sBuf = "delete from workspace_parameters " & _
            "where parameter_type = " & CStr(gintParameterConnect)
    dbFile.Execute sBuf, dbFailOnError

    ' Create the built-in parameter, default directory, for each workspace in the db
    Call InsertBuiltInParameter(dbFile, PARAM_DEFAULT_DIR,
gstrEmptyString, PARAM_DEFAULT_DIR_DESC)

    Call UpdateDbDtls(dbFile, sVersion)

    Exit Sub

Upgrade23to24Err:
    UpgradeWsp.Rollback
    Call LogErrors(Errors)
    Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
           LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade243to25(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

    Dim sBuf As String
    Dim qy As DAO.QueryDef
    Dim rTemp As DAO.Recordset
    Dim lId As Long
    Dim cTempStr As New cStringSM
    On Error GoTo Upgrade243to25Err

    sBuf = "Release " & gsVersion25 & " onwards, new 'Connections' must be
created for all " & _
           "connection strings. " & vbCrLf & vbCrLf & _
           "Connections will appear under the Globals/Connections " & _
           "node in the workspace. " & vbCrLf & _
           "A list of all 'Connections' (instead of 'Connection Strings') " & _
           "in the workspace will be displayed in the 'Connections' field for " & _
           "ODBC steps on the Step definition screen. " & vbCrLf & vbCrLf & _
           "Each Connection can be marked as static or dynamic. " & vbCrLf & _
           "Dynamic connections will be created when a step starts execution and "
& _
           "closed once the step completes. " & vbCrLf & _
           "Static connections will be kept open till the run completes." & vbCrLf &
vbCrLf & _
           "Currently dynamic 'Connections' have been created for all existing
'Connection Strings' " & _
           "with the suffix " & CONNECTION_STRINGS_TO_NAME_SUFFIX
    Call MsgBox(sBuf, vbOKOnly + vbApplicationModal, "Upgrade database")

    ' Add a new table for the connection name entity
    ' This table has been added in order to satisfy the TPC-H requirement that
    ' all the queries in a stream need to be executed on a single connection.
    sBuf = CreateConnectionDtlsTableScript()
    dbFile.Execute sBuf, dbFailOnError

    ' Add an identifier column for the connection_name_id field
    sBuf = "alter table att_identifiers " & _
           " add column " & FLD_ID_CONN_NAME & " long "
    dbFile.Execute sBuf, dbFailOnError

    Call UpdateDbDtls(dbFile, sVersion)

    ' insert connection_dtl records for each of the connection strings
    sBuf = "select * from " & TBL_CONNECTION_STRINGS
    Set rTemp = dbFile.OpenRecordset(sBuf, dbOpenSnapshot)

    sBuf = "insert into " & TBL_CONNECTION_DTLS & _
            "( " & FLD_ID_WORKSPACE & _
            ", " & FLD_ID_CONN_NAME & _
            ", " & FLD_CONN_DTL_CONNECTION_NAME & _
            ", " & FLD_CONN_DTL_CONNECTION_STRING & _
            ", " & FLD_CONN_DTL_CONNECTION_TYPE & " ) " & _
           " values ( [w_id], [c_id], [c_name], [c_str], [c_type] ) "
    Set qy = dbFile.CreateQueryDef("", sBuf)

    lId = glMinId
    If rTemp.RecordCount <> 0 Then
        rTemp.MoveFirst

        While Not rTemp.EOF
           qy.Parameters("w_id").Value = rTemp.Fields(FLD_ID_WORKSPACE)
           qy.Parameters("c_id").Value = lId
           qy.Parameters("c_name").Value =
rTemp.Fields(FLD_CONN_STR_CONNECTION_NAME) &
CONNECTION_STRINGS_TO_NAME_SUFFIX
           qy.Parameters("c_str").Value =
rTemp.Fields(FLD_CONN_STR_CONNECTION_NAME)
           qy.Parameters("c_type").Value = ConnTypeDynamic

           qy.Execute dbFailOnError

           lId = lId + 1
           rTemp.MoveNext
        Wend
    End If
    qy.Close
    rTemp.Close

    ' Initialize the value of the connection_name_id
    sBuf = "update att_identifiers " & _
           " set " & FLD_ID_CONN_NAME & " = " & Str(lId)
    dbFile.Execute sBuf, dbFailOnError
```

```
   ' Update the start_directory field in att_steps to point to the newly
   ' created connections
   Call ReadStepsInWorkspace(rTemp, qy, glInvalidId, dbLoad:=dbFile, _
         bSelectArchivedRecords:=False)

   sBuf = "update " & TBL_STEPS & _
      " set " & FLD_STEPS_EXEC_DTL & " = [c_name] " & _
      " where " & FLD_ID_STEP & " = [s_id] " & _
      " and " & FLD_STEPS_VERSION_NO & " = [ver_no] "
   Set qy = dbFile.CreateQueryDef("", sBuf)

   If rTemp.RecordCount <> 0 Then
      rTemp.MoveFirst

      While Not rTemp.EOF
         If rTemp.Fields(FLD_STEPS_EXEC_MECHANISM).Value =
gintExecuteODBC Then
            If Not (StringEmpty("" & rTemp.Fields(FLD_STEPS_EXEC_DTL)))
Then
               sBuf = rTemp.Fields(FLD_STEPS_EXEC_DTL)
               ' Strip the enclosing "%" characters
               sBuf = Mid(sBuf, 2, Len(sBuf) - 2) &
CONNECTION_STRINGS_TO_NAME_SUFFIX

               qy.Parameters("c_name").Value = sBuf
               qy.Parameters("s_id").Value = rTemp.Fields(FLD_ID_STEP)
               qy.Parameters("ver_no").Value =
rTemp.Fields(FLD_STEPS_VERSION_NO)

               qy.Execute dbFailOnError
            End If
         End If
         rTemp.MoveNext
      Wend
   End If

   qy.Close
   rTemp.Close

   Exit Sub

Upgrade243to25Err:
   UpgradeWsp.Rollback
   Call LogErrors(Errors)
   Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
         LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade25to251(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

   On Error GoTo Upgrade25to251Err

   ' Create the built-in parameters, run_id and output_dir, for each workspace in
the db
   Call InsertBuiltInParameter(dbFile, PARAM_RUN_ID, gstrEmptyString,
PARAM_RUN_ID_DESC)
   Call InsertBuiltInParameter(dbFile, PARAM_OUTPUT_DIR, gstrEmptyString,
PARAM_OUTPUT_DIR_DESC)

   Call UpdateDbDtls(dbFile, sVersion)

   Exit Sub

Upgrade25to251Err:
   UpgradeWsp.Rollback
   Call LogErrors(Errors)
   Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
         LoadResString(errUpgradeFailed)

End Sub
Private Sub Upgrade242to243(UpgradeWsp As DAO.Workspace, dbFile As
Database, sVersion As String)

   Dim sBuf As String
   Dim cTempStr As New cStringSM
   Dim iResponse As Integer

   On Error GoTo DeleteHistoryErr

   Call DeleteRunHistory(dbFile)

   On Error GoTo Upgrade242to243Err

   UpgradeWsp.CommitTrans

   UpgradeWsp.BeginTrans

   ' Add a parameter type field and a description field to the parameter table
   sBuf = "alter table run_step_details " & _
         " add column parent_instance_id LONG "

   dbFile.Execute sBuf, dbFailOnError

   sBuf = "alter table run_step_details " & _
         " add column iterator_value TEXT(255) "

   dbFile.Execute sBuf, dbFailOnError

   Call AlterFieldType(dbFile, TBL_RUN_STEP_DTLS, "start_time",
DATA_TYPE_CURRENCY)
   Call AlterFieldType(dbFile, TBL_RUN_STEP_DTLS, "end_time",
DATA_TYPE_CURRENCY)
   Call AlterFieldType(dbFile, TBL_RUN_STEP_HDR, "start_time",
DATA_TYPE_CURRENCY)
   Call AlterFieldType(dbFile, TBL_RUN_STEP_HDR, "end_time",
DATA_TYPE_CURRENCY)

   Call UpdateDbDtls(dbFile, sVersion)

   Exit Sub

DeleteHistoryErr:
   ' This is not a critical error - continue with upgrade
   Call LogErrors(Errors)
   Resume Next

Upgrade242to243Err:
   UpgradeWsp.Rollback
   Call LogErrors(Errors)
   Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
         LoadResString(errUpgradeFailed)

End Sub
'*****************************************************************
' The AlterFieldType Sub procedure requires three string
' parameters. The first string specifies the name of the table
' containing the field to be changed. The second string specifies
' the name of the field to be changed. The third string specifies
' the new data type for the field.
'*****************************************************************

Private Sub AlterFieldType(dbFile As Database, TblName As String, FieldName
As String, _
               NewDataType As String)
   Dim qdf As DAO.QueryDef
   Dim sSql As String

   ' Add a temporary field to the table.
   sSql = "ALTER TABLE [" & TblName & _
         "] ADD COLUMN AlterTempField " & NewDataType
   Set qdf = dbFile.CreateQueryDef("", sSql)
   qdf.Execute

   ' Copy the data from old field into the new field.
```

```
   qdf.SQL = "UPDATE DISTINCTROW [" & TblName & "] SET
AlterTempField = [" & FieldName & "]"
   qdf.Execute

   ' Delete the old field.
   qdf.SQL = "ALTER TABLE [" & TblName & "] DROP COLUMN [" &
FieldName & "]"
   qdf.Execute

   ' Rename the temporary field to the old field's name.
   dbFile.TableDefs("[" & TblName & "]").Fields("AlterTempField").Name =
FieldName
   dbFile.TableDefs.Refresh

   ' Clean up.
End Sub
Private Sub Upgrade01to21(UpgradeWsp As DAO.Workspace, dbFile As
DAO.Database, sVersion As String)
   Dim sSql As String

   On Error GoTo Upgrade01to21Err

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

   dbFile.Execute sSql, dbFailOnError

   sSql = "insert into db_details " & _
         "( db_version ) values ( '" & sVersion & "' ) "

   dbFile.Execute sSql, dbFailOnError

   Call UpdateContinuationCriteria(dbFile)

   Exit Sub

Upgrade01to21Err:
   Call LogErrors(Errors)
   UpgradeWsp.Rollback
   Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
         LoadResString(errUpgradeFailed)

End Sub
Private Function UpgradeDb(UpgradeWsp As DAO.Workspace, dbFile As
Database, _
      sVerTo As String, sVerFrom As String) As Boolean

   Dim sMsg As String

   On Error GoTo UpgradeDbErr

   UpgradeDb = False
   If Not ValidUpgrade(sVerTo, sVerFrom) Then Exit Function

   If NoDbChanges(sVerTo, sVerFrom) Then
      UpgradeDb = True
      Exit Function
   End If

   sMsg = "The database needs to be upgraded from Version " & sVerFrom & _
         " to Version " & sVerTo & "." & vbCrLf & _
         "Proceed?"
   If Not Confirm(Buttons:=vbYesNo, strMessage:=sMsg, strTitle:="Upgrade
database") Then
      Exit Function
   End If

   UpgradeWsp.BeginTrans

   Select Case sVerFrom
      Case gsVersion25
         Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

      Case gsVersion243
         Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
         Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

      Case gsVersion24, gsVersion241, gsVersion242
         sMsg = "After this upgrade, the run history for previous runs will no
longer be available. " & _
               "Continue?"
         If Not Confirm(Buttons:=vbYesNo, strMessage:=sMsg,
strTitle:="Upgrade database") Then
            UpgradeWsp.CommitTrans
            Exit Function
         End If
         Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion243)
         Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
         Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

      Case gsVersion23
         Call Upgrade23to24(UpgradeWsp, dbFile, gsVersion24)
         Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion242)
         Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
         Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

      Case gsVersion21
         Call Upgrade21to23(UpgradeWsp, dbFile, gsVersion23)
         Call Upgrade23to24(UpgradeWsp, dbFile, gsVersion24)
         Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion242)
         Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
         Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

      Case gsVersion10
         Call Upgrade10to21(UpgradeWsp, dbFile, gsVersion21)
         Call Upgrade21to23(UpgradeWsp, dbFile, gsVersion23)
         Call Upgrade23to24(UpgradeWsp, dbFile, gsVersion24)
         Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion242)
         Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
         Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

      Case gsVersion01
         Call Upgrade01to21(UpgradeWsp, dbFile, gsVersion21)
         Call Upgrade21to23(UpgradeWsp, dbFile, gsVersion23)
         Call Upgrade23to24(UpgradeWsp, dbFile, gsVersion24)
         Call Upgrade242to243(UpgradeWsp, dbFile, gsVersion242)
         Call Upgrade243to25(UpgradeWsp, dbFile, gsVersion25)
         Call Upgrade25to251(UpgradeWsp, dbFile, gsVersion251)

   End Select

   UpgradeWsp.CommitTrans

   UpgradeDb = True
   Exit Function

UpgradeDbErr:
   Call LogErrors(Errors)
   ShowError errUpgradeFailed

End Function
Private Function DBVersion(TestDb As Database) As String
   ' Retrieves the database version
   Dim rVersion As Recordset

   On Error GoTo DBVersionErr

   Set rVersion = TestDb.OpenRecordset("Select db_version from db_details ", _
         dbOpenForwardOnly)

   BugAssert rVersion.RecordCount <> 0
   DBVersion = rVersion!db_version

   rVersion.Close
   Exit Function

DBVersionErr:
```

```
    If Err.Number = merrDaoTableMissing Then
        DBVersion = gsVersion01
    Else
        LogErrors Errors
        Err.Raise vbObjectError + errUpgradeFailed, mstrModuleName, _
                LoadResString(errUpgradeFailed)
    End If

End Function

Private Function ValidUpgrade(sVerTo As String, sVerFrom As String) As
Boolean

    If sVerTo = gsVersion And sVerFrom = gsVersion251 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion25 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion243 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion242 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion241 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion24 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion23 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion21 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion10 Then
        ValidUpgrade = True
    ElseIf sVerTo = gsVersion And sVerFrom = gsVersion01 Then
        ValidUpgrade = True
    Else
        ValidUpgrade = False
    End If

End Function
```

DebugSM.bas

```
Attribute VB_Name = "DebugSM"
'  FILE:      DebugSM.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Contains all the functions that carry out error/debug
'             processing for the project.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
' Most of the functions in this module that manipulate the
' error object do not have an On Error GoTo statement - this
' is because it will clear the passed in error object - let
' the calling functions handle the errors raised by this
' module, if any
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "DebugSM."

Private mcLogFile As cFileSM
Private mcErrorFile As cFileSM

Private Const FORMAT_MESSAGE_FROM_SYSTEM = &H1000
Private Const FORMAT_MESSAGE_IGNORE_INSERTS = &H200
Private Const pNull = 0

Declare Function FormatMessage Lib "kernel32" Alias "FormatMessageA"
(ByVal dwFlags As Long, lpSource As Any, ByVal dwMessageId As Long,
ByVal dwLanguageId As Long, ByVal lpbuffer As String, ByVal nSize As Long,
Arguments As Long) As Long
Public Function Confirm(Optional lngMessageCode As conConfirmMsgCodes, _
        Optional lngTitleCode As conConfirmMsgTitleCodes, _
        Optional TitleParameter As String, _
        Optional ByVal Buttons As Integer = -1, _
        Optional strMessage As String = gstrEmptyString, _
        Optional strTitle As String = gstrEmptyString) _
        As Boolean
    ' Displays a confirmation message corresponding to the
    ' passed in message code. Returns True if the user says
    ' Ok and False otherwise

    Dim intResponse As Integer
    Dim intButtonStyle As Integer

    On Error GoTo ConfirmErr

    Confirm = False

    ' If the buttons style hasn't been specified, set the
    ' default style to display OK and Cancel buttons
    If Buttons = -1 Then
        intButtonStyle = vbOKCancel
    Else
        intButtonStyle = Buttons
    End If

    ' Find the message string for the passed in code
    If StringEmpty(strMessage) Then
        strMessage = Trim$(LoadResString(lngMessageCode))
    End If

    If StringEmpty(strTitle) Then
        strTitle = Trim$(LoadResString(lngTitleCode))
    End If

    If Not StringEmpty(TitleParameter) Then
        strTitle = strTitle & Chr$(vbKeySpace) & _
                gstrSQ & TitleParameter & gstrSQ
    End If

    ' Display the confirmation message with the Cancel button
    ' set to the default - assume that we are confirming
    ' potentially dangerous operations!
    intResponse = MsgBox(strMessage, _
            intButtonStyle + vbQuestion + vbApplicationModal, _
            strTitle)

    ' Translate the user response into a True/False return code
    If intButtonStyle = vbOKCancel Then
        If intResponse = vbOK Then
            Confirm = True
        Else
            Confirm = False
        End If
    Else
        If intResponse = vbYes Then
            Confirm = True
        Else
            Confirm = False
        End If
    End If

    Exit Function

ConfirmErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    gstrSource = mstrModuleName & "Confirm"
    Err.Raise vbObjectError + errConfirmFailed, _
            gstrSource, _
            LoadResString(errConfirmFailed)
```

```
End Function
Public Sub LogSystemError()
   Dim eErrCode As Long

   eErrCode = GetLastError()
   If eErrCode <> 0 Then
      WriteToFile "System Error: " & eErrCode & vbCrLf & ApiError(eErrCode),
_
            blnError:=True
   End If

End Sub
Public Function ApiError(ByVal e As Long) As String

   Dim s As String
   Dim c As Long

   s = String(256, 0)
   c = FormatMessage(FORMAT_MESSAGE_FROM_SYSTEM Or _
         FORMAT_MESSAGE_IGNORE_INSERTS, _
         pNull, e, 0&, s, Len(s), ByVal pNull)
   If c Then ApiError = e & ": " & Left$(s, c)

End Function

' Output flags determine output destination of BugAsserts and messages
#Const afLogfile = 1
#Const afMsgBox = 2
#Const afDebugWin = 4
#Const afAppLog = 8

' Display appropriate error message, and then stop
' program.  These errors should NOT be possible in
' shipping product.
Sub BugAssert(ByVal fExpression As Boolean, _
            Optional sExpression As String)
#If afDebug Then
   If fExpression Then Exit Sub
   BugMessage "BugAssert failed: " & sExpression
   Stop
#End If
End Sub

Sub BugMessage(sMsg As String)

#If afDebug And afLogfile Then
   ' Since we are writing log messages, the error flag is turned off
   Call WriteToFile(sMsg, False)
#End If
#If afDebug And afMsgBox Then
   MsgBox sMsg
#End If
#If afDebug And afDebugWin Then
   Debug.Print sMsg
#End If
#If afDebug And afAppLog Then
   App.LogEvent sMsg
#End If

End Sub
Public Function ProjectLogFile() As String

   ProjectLogFile = mcLogFile.FileName

End Function
Public Function ProjectErrorFile() As String

   ProjectErrorFile = mcErrorFile.FileName

End Function

Private Sub WriteToFile(sMsg As String, Optional ByVal blnError As Boolean)
   ' Calls procedures to write the passed in message to the log -
   ' The blnError flag is used to indicate that the message
   ' should be logged to the error file - by default the log
   ' file is used

   Dim mcFileObj As cFileSM
   Dim strFileName As String
   Dim strFileHdr As String

   On Error GoTo WriteToFileErr

   If blnError Then
      If mcErrorFile Is Nothing Then
         Set mcErrorFile = New cFileSM
      End If
      Set mcFileObj = mcErrorFile
   Else
      If mcLogFile Is Nothing Then
         Set mcLogFile = New cFileSM
      End If
      Set mcFileObj = mcLogFile
   End If

   If StringEmpty(mcFileObj.FileName) Then
      If blnError Then
         strFileName = gstrProjectPath & "\" & App.EXEName & ".ERR"
         strFileHdr = "Stepmaster Errors"
      Else
         strFileName = gstrProjectPath & "\" & App.EXEName & ".DBG"
         strFileHdr = "Stepmaster Log"
      End If

      mcFileObj.FileName = strFileName
      mcFileObj.WriteLine strFileHdr
      mcFileObj.WriteLine "Log start time : " & Now
   End If

   mcFileObj.WriteLine sMsg

   Exit Sub

WriteToFileErr:
   ' Display the error code raised by Visual Basic
   Call DisplayErrors(Errors)
   ' An error message would've been displayed by the called
   ' procedures

End Sub
Public Sub WriteMessage(sMsg As String)

   Call WriteToFile(sMsg, True)

End Sub

Sub BugTerm()
#If afDebug And afLogfile Then
   ' Close log file
   mcLogFile.CloseFile
#End If
End Sub

Public Sub ShowError(ByVal ErrorCode As errErrorConstants, _
      Optional ByVal ErrorSource As String = gstrEmptyString, _
      Optional ByVal OptArgs As String = gstrEmptyString, _
      Optional ByVal DoWriteError As Boolean = True)

   If DoWriteError Then
      ' Call a procedure to write the error to a log file
      Call WriteError(ErrorCode, ErrorSource, OptArgs)
   End If

   ' Re-initialize the values of the Error object before
   ' displaying the error to the user
   Call InitErrObject(ErrorCode, ErrorSource, OptArgs)
```

```
   Call DisplayErrors(Errors)

   Err.Clear

End Sub
Public Sub WriteError(ByVal ErrorCode As errErrorConstants, _
      Optional ByVal ErrorSource As String = gstrEmptyString, _
      Optional ByVal OptArgs As String = gstrEmptyString)

   ' Initialize the values of the Error object before
   ' calling the log function
   Call InitErrObject(ErrorCode, ErrorSource, OptArgs)

   Call LogErrors(Errors)

   Err.Clear

End Sub
Private Sub InitErrObject(ByVal ErrorCode As errErrorConstants, _
      Optional ByVal ErrorSource As String = gstrEmptyString, _
      Optional ByVal OptArgs As String = gstrEmptyString)

   Dim lngError As Long

   lngError = IIf(ErrorCode > vbObjectError And ErrorCode < vbObjectError +
65535, _
         ErrorCode - vbObjectError, ErrorCode)
   Err.Number = lngError + vbObjectError
   Err.Description = LoadResString(lngError) & OptArgs
   Err.Source = App.EXEName & ErrorSource

End Sub
Public Sub ShowMessage(ByVal MessageCode As errErrorConstants, _
      Optional ByVal OptArgs As String)

   Dim strMessage As String

   On Error GoTo ShowMessageErr

   strMessage = LoadResString(MessageCode) & OptArgs

   ' Write the error to a log file
   BugMessage strMessage

   MsgBox strMessage, vbOKOnly

   Exit Sub

ShowMessageErr:
   ' Log the error and exit
   Call DisplayErrors(Errors)

End Sub
Public Sub ShowMessageStr(sMessage As String)

   ' Write the error to a log file
   BugMessage sMessage

   MsgBox sMessage, vbOKOnly

End Sub

Public Sub DisplayErrors(myErrCollection As Errors)
   Dim strError As String
   Dim errLoop As Error
   Dim errCode As Long

   ' Enumerate Errors collection and display properties of
   ' each Error object.
   If Err.Number <> 0 Then
      If Err.Number > vbObjectError And Err.Number < (vbObjectError + 65536)
Then
         errCode = Err.Number - vbObjectError
      Else
         errCode = Err.Number
      End If
      strError = "Error # " & Str(errCode) & " was generated by " _
         & Err.Source & Chr(13) & Err.Description
      MsgBox strError, , "Error", Err.HelpFile, Err.HelpContext
   Else
      For Each errLoop In myErrCollection
         With errLoop
            If Err.Number > vbObjectError And Err.Number < (vbObjectError +
65536) Then
               errCode = .Number - vbObjectError
            Else
               errCode = .Number
            End If
            strError = "Error #" & errCode & vbCrLf
            strError = strError & "    " & .Description & vbCrLf
            strError = strError & _
               "    (Source: " & .Source & ")" & vbCrLf
'           strError = strError & _
'              "Press F1 to see topic " & .HelpContext & vbCrLf
'           strError = strError & _
'              "    in the file " & .HelpFile & "."
         End With

         MsgBox strError
      Next
   End If

End Sub
Public Sub LogErrors(myErrCollection As Errors)
   Dim cColErrors As cVectorStr
   Dim strError As String
   Dim errLoop As Error
   Dim errCode As Long
   Dim lngIndex As Long

   Set cColErrors = New cVectorStr

   ' Enumerate Errors collection and display properties of
   ' each Error object.
   If Err.Number <> 0 Then
      If Err.Number > vbObjectError And Err.Number < (vbObjectError + 65536)
Then
         errCode = Err.Number - vbObjectError
      Else
         errCode = Err.Number
      End If
      strError = "Error # " & Str(errCode) & " was generated by " _
         & Err.Source & vbCrLf & Err.Description

      cColErrors.Add strError
   End If

   ' Log all database errors, if any
   For Each errLoop In myErrCollection
      With errLoop
         If Err.Number > vbObjectError And Err.Number < (vbObjectError +
65536) Then
            errCode = .Number - vbObjectError
         Else
            errCode = .Number
         End If
         strError = "Error #" & errCode & vbCrLf
         strError = strError & "    " & .Description & vbCrLf
         strError = strError & _
            "    (Source: " & .Source & ")" & vbCrLf
      End With

      cColErrors.Add strError
   Next

   ' We can have a error handler now that we have stored all
   ' errors away safely! - having an error handler before
```

```
    ' enumerating all the errors would have cleared the error
    ' collection
    On Error GoTo LogErrorsErr
    gstrSource = mstrModuleName & "LogErrors"

    For lngIndex = 0 To cColErrors.Count - 1
        strError = cColErrors(lngIndex)
        Debug.Print strError
        Call WriteToFile(strError, True)
    Next lngIndex

    Set cColErrors = Nothing

    Exit Sub

LogErrorsErr:
    ' Display the error code raised by Visual Basic
    DisplayErrors Errors
    On Error GoTo 0
    ShowError errUnableToWriteError, DoWriteError:=False

End Sub
```

ErrHandling.bas

```
Attribute VB_Name = "ErrHandling"
'  FILE:      ErrHandling.bas
'             Reporting Program for Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Error handling module for TPCHMetrics report generation
program
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Sub DisplayErrors()
    ' Display error information such as error number, description, etc to the user

    Dim strError As String
    Dim errLoop As DAO.Error
    Dim errCode As Long

    If Err.Number <> 0 Then
        ' Visual Basic Error
        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
        ' DAO Error
        ' Enumerate Errors collection and display properties of
        ' each Error object.
        For Each errLoop In DAO.Errors
            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
            MsgBox strError
        Next
    End If

End Sub
```

FileCommon.bas

```
Attribute VB_Name = "FileCommon"
'  FILE:      FileCommon.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    This module contains common functionality to display
'             the File Open dialog.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "FileCommon."

Private Enum EOpenFile
    OFN_OVERWRITEPROMPT = &H2
    OFN_HIDEREADONLY = &H4
    OFN_FILEMUSTEXIST = &H1000
    OFN_EXPLORER = &H80000
End Enum

' The locations for the different output files are presented to
' the user in a list box. These constants are used while loading the
' data and while reading the data from the list box.
' These constants also represent the different file types that are
' displayed to the user in File Open dialogs
Public Enum gFileTypes
    gintOutputFile = 0
'    gintLogFile = 1
    gintErrorFile
    gintStepTextFile
    gintOutputCompareFile
    gintDBFile
    gintDBFileNew
    gintImportFile
    gintExportFile
End Enum

Public Const gsSqlFileSuffix = ".sql"
Public Const gsCmdFileSuffix = ".cmd"

Public Const gsOutputFileSuffix = ".out"
Public Const gstrLogFileSuffix = ".log"
Public Const gsErrorFileSuffix = ".err"
Public Function BrowseDBFile() As String
    ' Prompts the user for a database file with the workspace information
    ' Call CallFileDialog to display the open file dialog
    BrowseDBFile = CallFileDialog(gintDBFile)

End Function
Public Function CallFileDialog(intFileType As Integer, _
        Optional ByVal strDefaultFile As String = gstrEmptyString) As String
    ' This function initializes the values of the filter property,
    ' the dialog title and flags for the File Open dialog depending
    ' on the FileType passed in
    ' It then calls ShowFileOpenDialog to set these properties and
    ' display the File Open dialog to the user

    ' All the properties used by the File Open dialog are defined
    ' as constants in this function and passed to ShowFileOpenDialog
    ' as parameters. So if any of the dialog properties need to be
```

```
    ' modified, these constants are what need to be changed
    Const s_DLG_TITLE_OPEN = "Open"
    Const s_DLG_TITLE_NEW = "New"
    Const s_DLG_TITLE_IMPORT = "Import From"
    Const s_DLG_TITLE_EXPORT = "Export To"

    Const mlng_FILE_STEP_TEXT_FLAGS = OFN_EXPLORER Or
OFN_FILEMUSTEXIST Or OFN_HIDEREADONLY
    Const mlng_FILE_OUTPUT_COMPARE_FLAGS =
mlng_FILE_STEP_TEXT_FLAGS
    Const mlng_FILE_DB_FLAGS = mlng_FILE_STEP_TEXT_FLAGS
    Const mlng_FILE_OUTPUT_FLAGS = OFN_EXPLORER Or
OFN_HIDEREADONLY Or OFN_OVERWRITEPROMPT
    Const mlng_FILE_LOG_FLAGS = mlng_FILE_OUTPUT_FLAGS
    Const mlng_FILE_ERROR_FLAGS = mlng_FILE_OUTPUT_FLAGS
    Const mlng_FILE_DB_NEW_FLAGS = mlng_FILE_OUTPUT_FLAGS

    Const mstr_FILE_ALL_FILTER = "|All Files (*.*)|*.*"
    Const mstr_FILE_STEP_TEXT_FILTER = "Query Files (*" & gsSqlFileSuffix
& _
        ")|*" & gsSqlFileSuffix & "|Command Script Files (*" & gsCmdFileSuffix
& _
        ")|*" & gsCmdFileSuffix
    Const mstr_FILE_OUTPUT_COMPARE_FILTER = "Text Files (*.txt)|*.txt"
    Const mstr_FILE_OUTPUT_FILTER = "Output Files (*.out)|*.out"
    Const mstr_FILE_LOG_FILTER = "Log Files (*.log)|*.log"
    Const mstr_FILE_ERROR_FILTER = "Error Files (*.err)|*.err"
    Const mstr_FILE_DB_FILTER = "Stepmaster Workspace Files (*" &
gsDefDBFileExt & ")|*" & gsDefDBFileExt

    Dim strFileName As String

    On Error GoTo CallFileDialogErr

    Select Case intFileType
      Case gintStepTextFile
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_STEP_TEXT_FILTER & mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_OPEN, _
          mlng_FILE_STEP_TEXT_FLAGS, _
          strDefaultFile)

      Case gintOutputCompareFile
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_OUTPUT_COMPARE_FILTER &
mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_OPEN, _
          mlng_FILE_OUTPUT_COMPARE_FLAGS, _
          strDefaultFile)

      Case gintOutputFile
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_OUTPUT_FILTER & mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_OPEN, _
          mlng_FILE_OUTPUT_FLAGS, _
          strDefaultFile)

'       Case gintLogFile
'         strFileName = ShowFileOpenDialog( _
'           mstr_FILE_LOG_FILTER & mstr_FILE_ALL_FILTER, _
'           s_DLG_TITLE_OPEN, _
'           mlng_FILE_LOG_FLAGS, _
'           strDefaultFile)
'
      Case gintErrorFile
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_ERROR_FILTER & mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_OPEN, _
          mlng_FILE_ERROR_FLAGS, _
          strDefaultFile)

      Case gintDBFile
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_DB_FILTER & mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_OPEN, _
          mlng_FILE_DB_FLAGS, _
          strDefaultFile)

      Case gintDBFileNew
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_DB_FILTER & mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_NEW, _
          mlng_FILE_DB_NEW_FLAGS, _
          strDefaultFile)

      Case gintImportFile
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_DB_FILTER & mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_IMPORT, _
          mlng_FILE_DB_FLAGS, _
          strDefaultFile)

      Case gintExportFile
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_DB_FILTER & mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_EXPORT, _
          mlng_FILE_DB_FLAGS, _
          strDefaultFile)

      Case Else
        BugAssert True, "Incorrect file type passed in."
        ' Default processing will be for the output file
        strFileName = ShowFileOpenDialog( _
          mstr_FILE_OUTPUT_FILTER & mstr_FILE_ALL_FILTER, _
          s_DLG_TITLE_OPEN, _
          mlng_FILE_OUTPUT_FLAGS, _
          strDefaultFile)

    End Select
    CallFileDialog = strFileName

    Exit Function

CallFileDialogErr:
    CallFileDialog = gstrEmptyString
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "CallFileDialog"
    Call ShowError(errBrowseFailed)

End Function
```

IteratorCommon.bas

```
Attribute VB_Name = "IteratorCommon"
'  FILE:      IteratorCommon.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Contains functionality common across StepMaster and
'             SMRunOnly, pertaining to iterators
'             Specifically, functions to read iterators records
'             in the workspace, load them in an array and so on.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "IteratorCommon."

Public Const gintMinIteratorSequence As Integer = 0

Public Sub RangeComplete(vntIterators As Variant)
    ' This is a debug procedure
```

```
    ' Checks if the from, to and step values are present in
    ' the array

    Dim bReset As Byte
    Dim bShift As Byte
    Dim lngIndex As Long

    ' Set the three lowest order bits to 1
    bReset = 7

    BugAssert IsArray(vntIterators) And Not IsEmpty(vntIterators), _
            "Iterators not specified!"

    For lngIndex = LBound(vntIterators) To _
            UBound(vntIterators)
        bShift = 1
        bShift = bShift * (2 ^ (vntIterators(lngIndex).IteratorType - 1))

        bReset = bReset Xor bShift
    Next lngIndex

    ' Assert that all the elements are present
    BugAssert bReset = 0, "Range not completely specified!"

End Sub
Public Sub LoadIteratorsForWsp(cStepsCol As cArrSteps, _
        ByVal lngWorkspaceId As Long, rstStepsInWsp As Recordset)
    ' Initializes the step records in with all the iterator
    ' values for each step

    Dim recIterators As Recordset

    On Error GoTo LoadIteratorsForWspErr

#If QUERY_ALL Then
    Dim dtStart As Date

    dtStart = Now
    Set recIterators = ReadWspIterators(lngWorkspaceId)

    Call LoadIteratorsArray(cStepsCol, recIterators)

    recIterators.Close

    BugMessage "QueryAll Read + load took: " & CStr(DateDiff("s", dtStart,
Now))

#Else
    Dim dtStart As Date
    Dim qyIt As DAO.QueryDef
    Dim sSql As String

    dtStart = Now
    If rstStepsInWsp.RecordCount = 0 Then
        Exit Sub
    End If

    ' This method has the advantage that if the steps are queried right, everything
else follows
    sSql = "Select step_id, version_no, type, iterator_value, " & _
        " sequence_no " & _
        " from iterator_values " & _
        " where step_id = [s_id] " & _
        " and version_no = [ver_no] "

    ' Order the iterators by sequence within a step
    sSql = sSql & " order by sequence_no "

    Set qyIt = dbsAttTool.CreateQueryDef(gstrEmptyString, sSql)
    rstStepsInWsp.MoveFirst

    While Not rstStepsInWsp.EOF

        qyIt.Parameters("s_id").Value = rstStepsInWsp!step_id
        qyIt.Parameters("ver_no").Value = rstStepsInWsp!version_no

        Set recIterators = qyIt.OpenRecordset(dbOpenSnapshot)

        Call LoadIteratorsArray(cStepsCol, recIterators)
        recIterators.Close

        rstStepsInWsp.MoveNext
    Wend

    qyIt.Close

    BugMessage "Query step at a time Read + load took: " & CStr(DateDiff("s",
dtStart, Now))

#End If

    Exit Sub

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

End Sub
Private Function ReadWspIterators(ByVal lngWorkspaceId As Long) As
Recordset

    ' This function will return a recordset that is populated
    ' with the iterators for all the steps in a given workspace

    Dim recIterators As Recordset
    Dim qyIt As DAO.QueryDef
    Dim strSql As String

    On Error GoTo ReadWspIteratorsErr
    gstrSource = mstrModuleName & "ReadWspIterators"

    strSql = "Select i.step_id, i.version_no, " & _
        " i.type, i.iterator_value, " & _
        " i.sequence_no " & _
        " from iterator_values i, att_steps a " & _
        " where i.step_id = a.step_id " & _
        " and i.version_no = a.version_no " & _
        " and a.workspace_id = [w_id] " & _
        " and a.archived_flag = [archived] "

    ' Find the highest X-component of the version number
    strSql = strSql & " AND cint( mid( a.version_no, 1, instr( a.version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS d " & _
        " WHERE a.step_id = d.step_id ) "

    ' Find the highest Y-component of the version number for the highest
X-component
    strSql = strSql & " AND cint( mid( a.version_no, instr( a.version_no, " &
gstrDQ & gstrVerSeparator & gstrDQ & " ) + 1 ) ) = " & _
        " ( select max( cint( mid( version_no, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) + 1 ) ) ) " & _
        " from att_steps AS b " & _
        " Where a.step_id = b.step_id " & _
        " AND cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) = " & _
        " ( select max( cint( mid( version_no, 1, instr( version_no, " & gstrDQ &
gstrVerSeparator & gstrDQ & " ) - 1 ) ) ) " & _
        " from att_steps AS c " & _
        " WHERE a.step_id = c.step_id ) ) "
```

```
    ' Order the iterators by sequence within a step
    strSql = strSql & " order by i.step_id, i.sequence_no "

    Set qyIt = dbsAttTool.CreateQueryDef(gstrEmptyString, strSql)
    qyIt.Parameters("w_id").Value = lngWorkspaceId
    qyIt.Parameters("archived").Value = False

    Set recIterators = qyIt.OpenRecordset(dbOpenSnapshot)

    qyIt.Close
    Set ReadWspIterators = recIterators

    Exit Function

ReadWspIteratorsErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errReadDataFailed, _
          gstrSource, LoadResString(errReadDataFailed)

End Function
Private Sub LoadIteratorsArray(cStepsCol As cArrSteps, _
       recIterators As Recordset)
    ' Initializes the step records with the iterators for
    ' the step

    Dim cNewIt As cIterator
    Dim cStepRec As cStep
    Dim lngStepId As Long

    On Error GoTo LoadIteratorsArrayErr
    gstrSource = mstrModuleName & "LoadIteratorsArray"

    If recIterators.RecordCount = 0 Then
       Exit Sub
    End If

    recIterators.MoveFirst
    While Not recIterators.EOF
       Set cNewIt = New cIterator

       lngStepId = CLng(ErrorOnNullField(recIterators, "step_id"))
       If Not cStepRec Is Nothing Then
          If cStepRec.StepId <> lngStepId Then
             Set cStepRec = cStepsCol.QueryStep(lngStepId)
          End If
       Else
          Set cStepRec = cStepsCol.QueryStep(lngStepId)
       End If

       ' Initialize iterator values
       cNewIt.IteratorType = CInt(ErrorOnNullField(recIterators, "type"))
       cNewIt.Value = CStr(ErrorOnNullField(recIterators, "iterator_value"))
       cNewIt.SequenceNo = CInt(ErrorOnNullField(recIterators, "sequence_no"))

       ' Add this record to the array of iterators
       cStepRec.LoadIterator cNewIt

       Set cNewIt = Nothing
       recIterators.MoveNext
    Wend

    Exit Sub

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

End Sub
```

MsgConfirm.bas

```
Attribute VB_Name = "MsgConfirm"
'  FILE:      MsgConfirm.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:   Contains constants for confirmation messages that
'             will be displayed by StepMaster
'  Contact:   Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' A public enum containing the codes for all the confirmation
' messages that will be used by the project - each of the codes
' has the prefix, con
Public Enum conConfirmMsgCodes
   conWspDelete = 2000
   conSave
   conStopRun
   conSaveConnect
   conSaveDB
End Enum

' A public enum containing the titles for all the confirmation
' messages that will be used by the project - each of the codes
' has the prefix, cont - most confirmation message codes will
' have a corresponding title code in here
Public Enum conConfirmMsgTitleCodes
   contWspDelete = 3000
   contSave
   contStopRun
   contSaveConnect
   contSaveDB
End Enum
```

OpenFiles.bas

```
Attribute VB_Name = "OpenFiles"
'  FILE:      OpenFiles.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:   This module holds a list of all files that have been
'             opened by the project. This module is needed since there
'             is no way to share static data between different instances
'             of a class.
'             Many procedure in this module do not do any error handling -
'             this is 'coz it is also used by procedures that log error
'             messages and any error handler will erase the collection
'             of errors!

'  Contact:   Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit


' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = ".OpenFiles."

Private mOpenFiles As cNodeCollections

Private Const mstrTempDir As String = "\Temp\"

' The maximum number of temporary files that we can create in a
```

```
' session
Private Const mlngMaxFileIndex As Long = 999999
Private Const mstrFileIndexFormat As String = "000000"
Private Const mstrTempFilePrefix As String = "SM"
Private Const mstrTempFileSuffix As String = ".cmd"

Private Const merrFileNotFound As Long = 76
Private Function GetFileHandle(strFileName) As cFileInfo

   Dim lngIndex As Long
   Dim blnFileOpen As Boolean

   If Not mOpenFiles Is Nothing Then

      blnFileOpen = False
      For lngIndex = 0 To mOpenFiles.Count - 1
         If mOpenFiles(lngIndex).FileName = strFileName Then
            blnFileOpen = True
            Exit For
         End If
      Next lngIndex

      If blnFileOpen Then
         Set GetFileHandle = mOpenFiles(lngIndex)
      Else
         Set GetFileHandle = Nothing
      End If
   Else
      Set GetFileHandle = Nothing
   End If

End Function

Private Function GetTempFileDir() As String

   Dim strTempFileDir As String

   On Error GoTo GetTempFileDirErr

   strTempFileDir = gstrProjectPath & mstrTempDir

   If StringEmpty(Dir$(strTempFileDir, vbDirectory)) Then
      MkDir strTempFileDir
   End If

   GetTempFileDir = strTempFileDir

   Exit Function

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

End Function
Public Function MakePathValid(strFileName As String) As String
   ' Checks if the passed in file path is valid

   Dim strFileDir As String
   Dim strTempDir As String
   Dim strTempFile As String
   Dim intPos As Integer
   Dim intStart As Integer

   On Error GoTo MakePathValidErr
   gstrSource = mstrModuleName & "MakePathValid"

   strTempFile = strFileName
   intPos = InstrR(strFileName, gstrFileSeparator)

   If intPos > 0 Then
      strFileDir = Left$(strTempFile, intPos - 1)
      If StringEmpty(Dir$(strFileDir, vbDirectory)) Then
         ' Loop through the entire path starting at the root
         ' since Mkdir can create only one level of sub-directory
         ' at a time
         intStart = InStr(strFileDir, gstrFileSeparator)

         Do While strTempDir <> strFileDir

            If intStart > 0 Then
               strTempDir = Left$(strFileDir, intStart - 1)
            Else
               strTempDir = strFileDir
            End If

            If StringEmpty(Dir$(strTempDir, vbDirectory)) Then
               ' If the specified directory doesn't exist, try to
               ' create it.
               MkDir strTempDir
            Else
               ' The directory exists - go to it's sub-directory
            End If
            intStart = InStr(intStart + 1, strFileDir, gstrFileSeparator)
         Loop

         ' Sanity check
         If StringEmpty(Dir$(strFileDir, vbDirectory)) Then
            ' We were unable to create the file directory
            ShowError errCreateDirectoryFailed, gstrSource, _
                  strFileDir, DoWriteError:=False
            MakePathValid = gstrEmptyString
         Else
            MakePathValid = strTempFile
         End If
      Else
         ' The specified directory exists - we should be able
         ' to create the output file in it
         MakePathValid = strTempFile
      End If
   Else
      ' The user has only specified a filename - VB will try
      ' to create it in the current directory
      MakePathValid = strTempFile
   End If

   Exit Function

MakePathValidErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   gstrSource = mstrModuleName & "MakePathValid"
   ' Log the filename for debug
   Call WriteError(errInvalidFile, gstrSource, strTempFile)
   On Error GoTo 0
   Err.Raise vbObjectError + errProgramError, gstrSource, _
         LoadResString(errProgramError)

End Function
Public Function OpenFileSM(strFileName As String) As Integer
   Dim intHFile As Integer
   Dim NewFileInfo As cFileInfo

   On Error GoTo OpenFileSMErr
   gstrSource = mstrModuleName & "OpenFileSM"

   If StringEmpty(strFileName) Then
      On Error GoTo 0
      Err.Raise vbObjectError + errInvalidFile, gstrSource, _
            LoadResString(errInvalidFile)
   End If

   If mOpenFiles Is Nothing Then
      Set mOpenFiles = New cNodeCollections
   End If
```

```
    Set NewFileInfo = GetFileHandle(strFileName)

    If NewFileInfo Is Nothing Then
        ' The file has not been opened yet

        ' If the filename has not been initialized, do not
        ' attempt to open it
        strFileName = MakePathValid(strFileName)

        If strFileName <> gstrEmptyString Then
            intHFile = FreeFile
            Open strFileName For Output Shared As intHFile

            Set NewFileInfo = New cFileInfo
            NewFileInfo.FileHandle = intHFile
            NewFileInfo.FileName = strFileName
            mOpenFiles.Load NewFileInfo
        Else
            ' Either the directory was invalid or s'thing failed
            ' Display the error to the user instead of trying
            ' to log to the file
            ShowError errInvalidFile, gstrSource, strFileName, _
                DoWriteError:=False
            intHFile = 0
        End If
    Else
        intHFile = NewFileInfo.FileHandle
    End If

    OpenFileSM = intHFile

    Exit Function

OpenFileSMErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    ' The Open command failed for some reason - write an error
    ' and let the calling function handle the error
    ShowError errInvalidFile, gstrSource, strFileName, _
        DoWriteError:=False
    OpenFileSM = 0

End Function
Public Function CreateTempFile() As String

    Dim strTempFileDir As String
    Dim strTempFileName As String

    Static lngLastFileIndex As Long

    On Error GoTo CreateTempFileErr

    strTempFileDir = GetTempFileDir()

    Do
        If lngLastFileIndex = mlngMaxFileIndex Then
            On Error GoTo 0
            Err.Raise vbObjectError + errMaxTempFiles, gstrSource, _
                LoadResString(errMaxTempFiles)
        End If

        lngLastFileIndex = lngLastFileIndex + 1
        strTempFileName = mstrTempFilePrefix & _
            Format$(lngLastFileIndex, mstrFileIndexFormat) & _
            mstrTempFileSuffix

        If Not StringEmpty(Dir$(strTempFileDir & strTempFileName)) Then
            ' Remove any files left over from a previous run,
            ' if they still exist
            Kill strTempFileDir & strTempFileName
        End If

    ' Looping in case the file delete doesn't go through for
    ' some reason
    Loop While Not StringEmpty(Dir$(strTempFileDir & strTempFileName))

    CreateTempFile = GetShortName(strTempFileDir)
    CreateTempFile = CreateTempFile & strTempFileName

    Exit Function

CreateTempFileErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = gstrSource & "CreateTempFile"
    On Error GoTo 0
    Err.Raise vbObjectError + errProgramError, gstrSource, _
        LoadResString(errProgramError)

End Function
Public Sub CloseFileSM(strFileName As String)
    Dim FileToClose As cFileInfo

    If Not mOpenFiles Is Nothing Then

        ' Get the handle to the open file, if it exists
        Set FileToClose = GetFileHandle(strFileName)

        If Not FileToClose Is Nothing Then
            Close FileToClose.FileHandle

            ' Remove the file info from the collection of open files
            mOpenFiles.Unload FileToClose.Position
        End If
    End If

End Sub
Public Sub CloseOpenFiles()
    Dim lIndex As Long

    If Not mOpenFiles Is Nothing Then
        For lIndex = mOpenFiles.Count - 1 To 0
            CloseFileSM (mOpenFiles(lIndex).FileName)
        Next lIndex
    End If

End Sub
```

ParameterCommon.bas

```
Attribute VB_Name = "ParameterCommon"
'  FILE:      ParameterCommon.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:   Contains functionality common across StepMaster and
'             SMRunOnly, pertaining to parameters
'             Specifically, functions to load parameter records
'             in an array.
'  Contact:   Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "ParameterCommon."

Public Sub LoadRecordsetInParameterArray(rstWorkSpaceParameters As
Recordset, _
    cParamCol As cArrParameters)

    Dim cNewParameter As cParameter

    On Error GoTo LoadRecordsetInParameterArrayErr
```

```
    If rstWorkSpaceParameters.RecordCount = 0 Then
        Exit Sub
    End If

    rstWorkSpaceParameters.MoveFirst
    While Not rstWorkSpaceParameters.EOF

        Set cNewParameter = New cParameter

        ' Initialize parameter values
        cNewParameter.ParameterId = rstWorkSpaceParameters.Fields(0)

        ' Call a procedure to raise an error if mandatory fields are
        ' null.
        cNewParameter.ParameterName = CStr( _
                ErrorOnNullField(rstWorkSpaceParameters, "parameter_name"))
        cNewParameter.ParameterValue = CheckForNullField( _
                rstWorkSpaceParameters, "parameter_value")
        cNewParameter.WorkspaceId = CStr( _
                ErrorOnNullField(rstWorkSpaceParameters, FLD_ID_WORKSPACE))
        cNewParameter.ParameterType = CStr( _
                ErrorOnNullField(rstWorkSpaceParameters, "parameter_type"))
        cNewParameter.Description = CheckForNullField( _
                rstWorkSpaceParameters, "description")

        cParamCol.Load cNewParameter

        Set cNewParameter = Nothing
        rstWorkSpaceParameters.MoveNext
    Wend

    Exit Sub

LoadRecordsetInParameterArrayErr:
    LogErrors Errors
    gstrSource = mstrModuleName & "LoadRecordsetInParameterArray"
    On Error GoTo 0
    Err.Raise vbObjectError + errLoadRsInArrayFailed, gstrSource, _
            LoadResString(errLoadRsInArrayFailed)
End Sub
```

Public.bas

```
Attribute VB_Name = "Public"
'  FILE:      Public.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    This module contains all the public constants for this project
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Const gsVersion01 As String = "0.1"
Public Const gsVersion10 As String = "1.0"
Public Const gsVersion21 As String = "2.1"
Public Const gsVersion23 As String = "2.3"
Public Const gsVersion24 As String = "2.4"
Public Const gsVersion241 As String = "2.4.1"
Public Const gsVersion242 As String = "2.4.2"
Public Const gsVersion243 As String = "2.4.3"
Public Const gsVersion25 As String = "2.5"
Public Const gsVersion251 As String = "2.5.1"
Public Const gsVersion253 As String = "2.5.3"
Public Const gsVersion254 As String = "2.5.4"
Public Const gsVersion255 As String = "2.5.5"
Public Const gsVersion As String = gsVersion255

' The same form is used for the creation of new nodes and
' updates to existing nodes (where each node can be a parameter,
' global step, etc.) A tag is set on each flag is used to indicate
' whether it is being called in the insert or update mode. The
' constants for these modes are defined below
Public Const gstrInsertMode = "Insert"
Public Const gstrUpdateMode = "Update"
Public Const gstrPropertiesMode = "View"

Public Const gstrEmptyString = ""
Public Const gstrSQ = "'"
Public Const gstrDQ = """"
Public Const gstrVerSeparator = "."
Public Const gstrBlank = " "

' Constants used to indicate type of node being processed
' The constants for the different objects correspond to the
' indexes in the menu control arrays (for both the main and popup
' menus) that are used to create new objects. That way we can
' use the index passed in by the click event to determine the
' type of node being processed
Public Const gintWorkspace = 1

' Decided to leave it here after some debate over whether it
' actually belongs in the cStep class definition
Public Enum gintStepType
    gintGlobalStep = 3
    gintManagerStep
    gintWorkerStep
End Enum

Public Const gintRunManager = 6
Public Const gintRunWorker = 7

Public Enum gintParameterNodeType
    gintParameter = 8
    gintNodeParamConnection
    gintNodeParamExtension
    gintNodeParamBuiltIn
End Enum

' Leave some constants free for newer types of parameters (?)
Public Const gintConnectionDtl = 15

Public Enum gintLabelNodeType
    gintGlobalsLabel = 21
    gintParameterLabel
    gintParamConnectionLabel
    gintParamExtensionLabel
    gintParamBuiltInLabel
    gintConnDtlLabel
    gintGlobalStepLabel
    gintStepLabel
End Enum

Public Enum ConnectionType
    ConnTypeStatic = 1
    ConnTypeDynamic
End Enum

Public Const giDefaultConnType As Integer = ConnTypeStatic

' The constants defined below are used to identify the different
' tabs. If any more step properties and thereby tabs are added
' to the tabbed dialog on the Step Properties form, they should
' be defined here and accessed in the code only using these
' pre-defined constants
' Note: These constants will mainly be used by the functions that
' initialize, customize and display the Step Properties form
Public Const gintDefinition = 0
Public Const gintExecution = 1
Public Const gintMgrDefinition = 2
Public Const gintPreExecutionSteps = 3
Public Const gintPostExecutionSteps = 4
Public Const gintFileLocations = 5
```

```
' These constants correspond to the index values in the imagelist
' associated with the tree view control. The imagelist contains
' the icons that will be displayed for each node.
Public Enum TreeImages
   gintImageWorkspaceClosed = 1
   gintImageWorkspaceOpen
   gintImageLabelClosed
   gintImageLabelOpen
   gintImageManagerClosedDis
   gintImageManagerClosedEn
   gintImageManagerOpenDis
   gintImageManagerOpenEn
   gintImageWorkerDis
   gintImageWorkerEn
   gintImageGlobalClosed
   gintImageGlobalOpen
   gintImageParameter
   gintImageRun
   gintImagePending
   gintImageStop
   gintImageDisabled
   gintImageAborted
   gintImageFailed
End Enum

' Public variable used to indicate the name of the function
' that raises an error
Public gstrSource As String

' Public instances of the different collections
Public gcParameters As cArrParameters
Public gcSteps As cArrSteps
Public gcConstraints As cArrConstraints
Public gcConnections As cConnections
Public gcConnDtls As cConnDtls

' Public constants for the index values of the different toolbar
' options. Will be used while dynamically enabling/disabling
' these options.
Public Const tbNew = 1
Public Const tbOpen = 2
Public Const tbSave = 3

Public Const tbCut = 5
Public Const tbCopy = 6
Public Const tbPaste = 7
Public Const tbDelete = 8

Public Const tbProperties = 10
Public Const tbRun = 11
Public Const tbStop = 12

' The initial version #
Public Const gstrMinVersion As String = "0.0"
Public Const gstrGlobalParallelism As String = "0"
Public Const gintMinParallelism As Integer = 1
Public Const gintMaxParallelism As Integer = 100

' Constant for the minimum identifier, used for all identifier, viz.
' step, workspace, etc.
Public Const glMinId As Long = 1
Public Const glInvalidId As Long = -1

' A parameter that has a special meaning to Stepmaster
' The system time will be substituted wherever it occurs
' (typically as a part of the error, log ... file names
Public Const gstrTimeStamp As String = "TIMESTAMP"
Public Const gstrEnvVarSeparator = "%"
Public Const gstrFileSeparator = "\"
Public Const gstrUnderscore = "_"

' Constants used by date and time formatting functions
Public Const gsTimeSeparator = ":"
Public Const gsDateSeparator = "-"
Public Const gsMsSeparator = "."
Public Const gsDtFormat = "00"
Public Const gsYearFormat = "0000"
Public Const gsTmFormat = "00"
Public Const gsMSecondFormat = "000"

' Default nothing value for a date variable
Public Const gdtmEmpty As Currency = 0

Public Const FMT_WSP_LOG_FILE As String = "yyyymmdd-hhnnss"

Public gsContCriteria() As String
' Note: Update the initialization of gsExecutionStatus in Initialize() if the
' InstanceStatus values are modified - also the boundary checks
Public gsExecutionStatus() As String

Public Const gsConnTypeStatic As String = "Static"
Public Const gsConnTypeDynamic As String = "Dynamic"

#If RUN_ONLY Then
Public Const gsCaptionRunWsp As String = "Run Workspace"
#End If

' Valid operations on a cNode object
Public Enum Operation
   QueryOp = 1
   InsertOp = 2
   UpdateOp = 3
   DeleteOp = 4
End Enum
```

RunCommon.bas

```
Attribute VB_Name = "RunCommon"
'  FILE:      RunCommon.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:   Contains common functions that are used during the execution
'             of a workspace.
'  Contact:   Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = ".RunCommon."


Public Function GetInstanceItValue(cInstanceRec As cInstance) As String

   ' Returns the iterator value for the instance, if an
   ' iterator has been defined for it
   Dim cStepIt As cRunColIt
   Dim cRunIterator As cRunItNode

   On Error GoTo GetInstanceItValueErr

   ' Since we create a dummy instance for Disabled and Pending steps,
   ' doesn't make sense to look at their iterators
   If cInstanceRec.Status <> gintDisabled And cInstanceRec.Status <>
gintPending Then
      Set cStepIt = cInstanceRec.Iterators

      If Not StringEmpty(cInstanceRec.Step.IteratorName) Then
         If cStepIt.Count > 0 Then
            Set cRunIterator = cStepIt(0)
            BugAssert cRunIterator.IteratorName =
cInstanceRec.Step.IteratorName, _
                  "The first iterator in the collection is the " & _
                  "one that has been defined for the step."
```

```
            If cRunIterator.IteratorName = cInstanceRec.Step.IteratorName Then
                GetInstanceItValue = cRunIterator.Value
            Else
                GetInstanceItValue = gstrEmptyString
            End If
        Else
            GetInstanceItValue = gstrEmptyString
        End If
    Else
        GetInstanceItValue = gstrEmptyString
    End If
Else
    GetInstanceItValue = gstrEmptyString
End If

Exit Function

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

End Function
```

RunInstHelper.bas

```
Attribute VB_Name = "RunInstHelper"
'  FILE:      RunInstHelper.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    This module contains helper procedures that are called by
'             cRunInst.cls
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this class
Private Const mstrModuleName As String = "RunInstHelper."

' Should be equal to the number of steps defined in cRunInst.cls
Public Const glngNumConcurrentProcesses As Long = 99
Public Const gintBitsPerByte = 8
Public Function AnyStepRunning(cFreeSteps As cVectorLng, arrFree() As Byte)
As Boolean

    Dim lngIndex As Long
    Dim intPosInByte As Integer
    Dim lngTemp As Long

    ' Check if there are any running instances to wait for
    If cFreeSteps.Count <> glngNumConcurrentProcesses Then

        ' For every free step, reset the corresponding element
        ' in the byte array to 0
        For lngIndex = 0 To cFreeSteps.Count - 1

            lngTemp = cFreeSteps(lngIndex) \ gintBitsPerByte
            intPosInByte = cFreeSteps(lngIndex) Mod gintBitsPerByte

            arrFree(lngTemp) = arrFree(lngTemp) Xor 2 ^ intPosInByte
        Next lngIndex

        AnyStepRunning = False

        ' Check if we have a non-zero bit in the byte array
        For lngIndex = LBound(arrFree) To UBound(arrFree) Step 1
            If arrFree(lngIndex) <> 0 Then
                ' We are waiting for a step to complete
                AnyStepRunning = True
                Exit For
            End If
        Next lngIndex

    Else
        AnyStepRunning = False
    End If

End Function


Public Function OrderConstraints(vntTempCons() As Variant, _
        intConsType As ConstraintType) As Variant
    ' Returns a variant containing all the constraint records in the order
    ' in which they should be executed

    Dim vntTemp As Variant
    Dim lngOuter As Long
    Dim lngInner As Long
    Dim cTempConstraint As cConstraint
    Dim cConstraints() As cConstraint
    Dim lngConsCount As Long
    Dim lngLbound As Long
    Dim lngUbound As Long
    Dim lngStep As Long

    On Error GoTo OrderConstraintsErr

    If intConsType = gintPreStep Then
        ' Since we are travelling up and we need to execute the constraints
        ' for the top-level steps first, reverse the order that they
        ' have been stored in the array
        lngLbound = UBound(vntTempCons)
        lngUbound = LBound(vntTempCons)
        lngStep = -1
    Else
        lngLbound = LBound(vntTempCons)
        lngUbound = UBound(vntTempCons)
        lngStep = 1
    End If

    lngConsCount = 0

    For lngOuter = lngLbound To lngUbound Step lngStep
        vntTemp = vntTempCons(lngOuter)

        If Not IsEmpty(vntTemp) Then
            ' Each of the elements is an array
            For lngInner = LBound(vntTemp) To UBound(vntTemp) Step 1
                If Not IsEmpty(vntTemp(lngInner)) Then
                    Set cTempConstraint = vntTemp(lngInner)

                    If Not cTempConstraint Is Nothing Then
                        ReDim Preserve cConstraints(lngConsCount)
                        Set cConstraints(lngConsCount) = cTempConstraint
                        lngConsCount = lngConsCount + 1
                    End If
                End If
            Next lngInner
        End If
    Next lngOuter

    ' Set the return value of the function to the array of
    ' constraints that has been built above
    If lngConsCount = 0 Then
        OrderConstraints = Empty
    Else
        OrderConstraints = cConstraints()
    End If

    Exit Function
```

```
OrderConstraintsErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errExecInstanceFailed, _
         mstrModuleName, LoadResString(errExecInstanceFailed)

End Function
```

ShellSM.bas

```
Attribute VB_Name = "ShellSM"
'  FILE:      ShellSM.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    This module contains a function that creates a process and
'             waits for it to complete.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Function SyncShell(CommandLine As String, Optional Timeout As Long,
_
   Optional WaitForInputIdle As Boolean) As Boolean

   Dim proc As PROCESS_INFORMATION
   Dim Start As STARTUPINFO
   Dim ret As Long
   Dim nMilliseconds As Long

   BugMessage "Executing: " & CommandLine
   If Timeout > 0 Then
      nMilliseconds = Timeout
   Else
      nMilliseconds = INFINITE
   End If

   'Initialize the STARTUPINFO structure:
   Start.cb = Len(Start)
   Start.dwFlags = STARTF_USESHOWWINDOW
   Start.wShowWindow = SW_SHOWMINNOACTIVE

   'Start the shelled application:
   CreateProcessA 0&, CommandLine, 0&, 0&, 1&, _
      NORMAL_PRIORITY_CLASS, 0&, 0&, Start, proc

   If WaitForInputIdle Then
      'Wait for the shelled application to finish setting up its UI:
      ret = InputIdle(proc.hProcess, nMilliseconds)
   Else
      'Wait for the shelled application to terminate:
      ret = WaitForSingleObject(proc.hProcess, nMilliseconds)
   End If

   CloseHandle proc.hProcess

   'Return True if the application finished. Otherwise it timed out or erred.
   SyncShell = (ret = WAIT_OBJECT_0)
End Function
```

SMErr.bas

```
Attribute VB_Name = "SMErr"
'  FILE:      SMErr.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    This module contains error code for all the errors that are
'             raised by StepMaster.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' A public enum containing the codes for all the error
' messages that will be displayed by the project - each
' of the codes has the prefix, err
Public Enum errErrorConstants
   errParameterIdInvalid = 1000
   errParameterNameMandatory
   errParameterInsertFailed
   errStepLabelOrTextOrFileRequired
   errMandatoryNodeTextMissing
   errParameterUpdateFailed
   errDupConnDtlName
   errDummy14
   errContCriteriaMandatory
   errContCriteriaNullForGlobal
   errContCriteriaInvalid = 1010
   errParamSeparatorMissing
   errStepTextOrTextFileMandatory
   errStepTextOrFile
   errEnabledFlagFalseForGlobal
   errEnabledFlagLetFailed
   errDegParallelismNullForGlobal
   errInvalidDegParallelism
   errExecutionMechanismInvalid
   errExecutionMechanismLetFailed
   errStepLevelNull = 1020
   errStepLevelZeroForGlobal
   errStepLevelLetFailed
   errRowCountNumeric
   errConnNameInvalid
   errTimeoutNumeric
   errResetConnPropertiesFailed
   errFailureThresholdNumeric
   errConnectionUpdateFailed
   errGlobalRunMethodMandatory
   errGlobalRunMethodNull = 1030
   errGlobalRunMethodInvalid
   errGlobalRunMethodLetFailed
   errNoTrueOption
   errGetOptionFailed
   errSetEnabled
   errStepLabelTextAndFileNull
   errInvalidNodeType
   errSetOptionFailed
   errQueryParameterFailed
   errParamNotFound = 1040
   errQueryStepFailed
   errStepNotFound
   errReadFromScreenFailed
   errCopyPropertiesToFormFailed
   errMandatoryFieldNull
   errUnableToCheckNull
   errUpgradeFailed
   errFindStepSequenceFailed
   errFindParentStepIdFailed
   errCircularReference = 1050
   errAddStepFailed
   errModifyStepFailed
   errDeleteColumnFailed
   errFindPositionFailed
   errDeleteStepFailed
   errRunExistsForStepFailed
   errCreateNewParentFailed
   errInsertNewStepVersionFailed
   errCreateNewStepFailed
   errDuplicateParameterName = 1060
   errCheckDupParameterNameFailed
   errConstraintTypeInvalid
```

errConstraintTypeLetFailed
errAddConstraintFailed
errWorkspaceIdMandatory
errInvalidWorkspaceData
errGetWorkspaceDetailsFailed
errNoWorkspaceLoaded
errWorkspaceAlreadyOpen
errDuplicateWorkspaceName = 1070
errWorkspaceNameMandatory
errWorkspaceNameSetFailed
errWorkspaceIdInvalid
errWorkspaceIdSetFailed
errWorkspaceInsertFailed
errWorkspaceDeleteFailed
errWorkspaceUpdateFailed
errInvalidFile
errCheckWorkspaceOpenFailed
errWriteFailed = 1080
errDeleteParameterRecordFailed
errUnableToLogOutput
errDeleteDBRecordFailed
errRunExistsForWorkspaceFailed
errClearHistoryFailed
errCreateDBFailed
errImportWspFailed
errStepModifyFailed
errStepDeleteFailed
errDummy3 = 1090
errUnableToGetWorkspace
errInvalidNode
errUnableToRemoveSubtree
errSetFileNameFailed
errStepTypeInvalid
errObjectMandatory
errBuiltInUpdateOnly
errInvalidStep
errTypeOfStepFailed
errGetParentKeyFailed = 1100
errLabelTextAndFileCheckFailed
errStepTextAndFileNull
errTextOrFileCheckFailed
errParentStepManager
errDeleteSubStepsFailed
errWorkspaceNameDuplicateFailed
errNewConstraintVersionFailed
errDeleteStepConstraintsFailed
errOldVersionMandatory
errLoadConstraintsInListFailed = 1110
errLoadGlobalStepsFailed
errDeleteConstraintFailed
errUpdateConstraintFailed
errConstraintIdInvalid
errConstraintIdSetFailed
errGlobalStepIdInvalid
errGlobalStepIdSetFailed
errUpdateVersionFailed
errQueryAdjacentConsFailed
errConstraintNotFound = 1120
errQueryConstraintFailed
errSetDBBeforeLoad
errLoadDataFailed
errLoadRsInArrayFailed
errConstraintsForStepFailed
errPreConstraintsForStepFailed
errPostConstraintsForStepFailed
errExecuteConstraintMethodFailed
errIdOrKeyMandatory
errInListFailed = 1130
errUnableToWriteChanges
errQuickSortFailed
errCheckParentValidFailed
errLogErrorFailed
errCopyListFailed
errConnected
errVersionMismatch
errStepNodeFailed
errWorkspaceSelectedFailed
errIdentifierSelectedFailed = 1140
errCheckForNullFieldFailed
errInstanceInUse
errSetVisiblePropertyFailed
errExportWspFailed
errMakeKeyValidFailed
errDummy16
errRunApplicationFailed
errStepLabelUnique
errDeleteSingleFile
errMakeIdentifierValidFailed = 1150
errTypeOfNodeFailed
errConstraintCommandFailed
errOpenDbFailed
errInsertNewConstraintsFailed
errLoadPostExecutionStepsFailed
errLoadPreExecutionStepsFailed
errCreateNewNodeFailed
errDeleteNodeFailed
errDisplayPopupFailed
errDisplayPropertiesFailed = 1160
errUnableToCreateNewObject
errDiffFailed
errLoadWorkspaceFailed
errTerminateProcessFailed
errCompareFailed
errCreateConnectionFailed
errShowFormFailed
errAbortFailed
errDeleteParameterFailed
errUpdateViewFailed = 1170
errParameterNewFailed
errCopyNodeFailed
errCutNodeFailed
errCheckObjectValidFailed
errDeleteViewNodeFailed
errMainFailed
errNewStepFailed
errProcessStepModifyFailed
errCustomizeStepFormFailed
errInitializeStepFormFailed = 1180
errInsertStepFailed
errIncVersionYFailed
errIncVersionXFailed
errShowCreateStepFormFailed
errShowStepFormFailed
errStepNewFailed
errUnableToApplyChanges
errUnableToCommitChanges
errGetStepNodeTextFailed
errSelectGlobalRunMethodFailed = 1190
errConnectionNameMandatory
errUpdateStepFailed
errBrowseFailed
errDummy4
errDummy1
errDummy2
errDummy
errUnableToPreviewFile
errCopyWorkspaceFailed
errCopyParameterFailed = 1200
errGetStepTypeAndPositionFailed
errCopyStepFailed
errMandatoryParameterMissing
errDeleteWorkspaceRecordsFailed
errCreateDirectoryFailed
errConfirmDeleteOrMoveFailed
errCreateWorkspaceFailed
errTypeOfObjectFailed
errCreateNodeFailed
errCreateParameterFailed = 1210

errInsertParameterFailed
errCreateStepFailed
errNoConstraintsCreated
errCopyFailed
errCloneFailed
errCloneGlobalFailed
errCloneWorkerFailed
errCloneManagerFailed
errLetStepTypeFailed
errUnableToCloseWorkspace = 1220
errUnableToModifyWorkspace
errUnableToCreateWorkspace
errAddArrayElementFailed
errUpdateSequenceFailed
errCannotCopySubSteps
errSubStepsFailed
errModifyInArrayFailed
errUpdateParentVersionFailed
errGetNodeTextFailed
errAddToArrayFailed = 1230
errDeleteFromArrayFailed
errQueryIndexFailed
errCreateNewConstraintVersionFailed
errGetRootNodeFailed
errPopulateWspDetailsFailed
errLoadRsInTreeFailed
errAddNodeToTreeFailed
errMaxTempFiles
errMoveFailed
errRootNodeKeyInvalid = 1240
errNextNodeFailed
errBranchWillMove
errMoveBranchInvalid
errCreateIdRecordsetFailed
errIdentifierColumnFailed
errGetIdentifierFailed
errGetStepTypeFailed
errUpdateConstraintSeqFailed
errDelParamsInWspFailed
errDuplicateConnectionName = 1250
errOpenWorkspaceFailed
errShowWorkspaceNewFailed
errShowWorkspaceModifyFailed
errPopulateListFailed
errExploreNodeFailed
errInitializeListNodeFailed
errMakeListColumnsFailed
errRefreshViewFailed
errExploreFailed
errCollapseNodeFailed = 1260
errUnableToProcessListViewClick
errSetEnabledForStepFailed
errDisplayStepFormFailed
errSetEnabledPropertyFailed
errInvalidDB
errDeleteConnectionFailed
errInvalidOperation
errLetOperationFailed
errIdGetFailed
errCommitFailed = 1270
errSaveParametersInWspFailed
errDeleteArrayElementFailed
errSaveWorkspaceFailed
errInitializeFailed
errLoadInArrayFailed
errSaveStepsInWspFailed
errCommitStepFailed
errStepIdGetFailed
errUnloadFromArrayFailed
errValidateFailed = 1280
errTextEnteredFailed
errStepLabelMandatory
errTextAndFileNullForManager
errFailureDetailsNullForMgr
errSetTabOrderFailed
errSaveWspConstraintsFailed
errCommitConstraintFailed
errUnloadStepConstraintsFailed
errUnableToModifyMenu
errConfirmFailed = 1290
errInitSubItemsFailed
errUpdateListNodeFailed
errAddNodeFailed
errLoadListNodeFailed
errAddListNodeFailed
errExecutionFailed
errSetListViewStyleFailed
errSetCheckedFailed
errGetCheckedFailed
errUnableToProcessListViewDblClick = 1300
errDefaultPosition
errShellFailed
errOpenFileFailed
errSetTBar97Failed
errConnectFailed
errApiFailed
errRegEntryInvalid
errParseStringFailed
errConstraintsForWspFailed
errPostConstraintsForWspFailed = 1310
errPreConstraintsForWspFailed
errLoadWspPostExecStepsFailed
errLoadWspPreExecStepsFailed
errLoadConstraintsOnFormFailed
errQueryFailed
errPasteNodeFailed
errShowAllWorkspacesFailed
errMakeFieldValidFailed
errInitializeTree
errRootNodeFailed = 1320
errDirectionInvalid
errUnableToDetListProperty
errUnableToGetListData
errItemNotFound
errItemDoesNotExist
errParamNameInvalid
errGetParamValueFailed
errSubValuesFailed
errStringOpFailed
errReadWorkspaceDataFailed = 1330
errUpdateRunDataFailed
errProgramError
errUnableToOpenFile
errLoadRunDataFailed
errExecuteODBCCommandFailed
errRunWorkspaceFailed
errExecuteStepFailed
errUnableToWriteError
errRunStepFailed
errSaveChanges = 1340
errDragDropFailed
errInvalidParameter
errAssignParametersFailed
errLoadLabelsInTreeFailed
errInstrRFailed
errInsertIteratorFailed
errDeleteIteratorFailed
errTypeInvalid
errLoadFailed
errDeleteFailed = 1350
errModifyFailed
errIteratorsFailed
errInsertFailed
errUpdateFailed
errDuplicateIterator
errSaveFailed
errReadDataFailed
errUnloadFailed

```
    errAddFailed
    errExecuteBranchFailed = 1360
    errRangeNumeric
    errRangeInvalid
    errNextStepFailed
    errUpdateDisplayFailed
    errDateToStringFailed
    errGetElapsedTimeFailed
    errMaxProcessesExceeded
    errInvalidProperty
    errInvalidChild
    errCreateInstanceFailed = 1370
    errInvalidForWorker
    errInstanceOpFailed
    errNavInstancesFailed
    errIterateFailed
    errExecInstanceFailed
    errDupIterator
End Enum
```

SortSM.bas

```
Attribute VB_Name = "SortSM"
'  FILE:      SortSM.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    This module contains an implementation of QuickSort.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Comment out for case-sensitive sorts
Option Compare Text

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "SortSM."

Private Function Compare(ByVal vntToCompare1 As Variant, _
        ByVal vntToCompare2 As Variant) As Integer

    On Error GoTo CompareErr

    Compare = 0

    If vntToCompare1.SequenceNo < vntToCompare2.SequenceNo Then
        Compare = -1
    ElseIf vntToCompare1.SequenceNo > vntToCompare2.SequenceNo Then
        Compare = 1
    End If

    Exit Function

CompareErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errCompareFailed, _
            gstrSource, _
            LoadResString(errCompareFailed)

End Function

Private Sub Swap(ByRef vntToSwap1 As Variant, _
        ByRef vntToSwap2 As Variant)

    Dim vntTemp As Variant

    On Error GoTo SwapErr

    If IsObject(vntToSwap1) And IsObject(vntToSwap2) Then
        Set vntTemp = vntToSwap1
        Set vntToSwap1 = vntToSwap2
        Set vntToSwap2 = vntTemp
    Else
        vntTemp = vntToSwap1
        vntToSwap1 = vntToSwap2
        vntToSwap2 = vntTemp
    End If

    Exit Sub

SwapErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errQuickSortFailed, mstrModuleName & "Swap", _
        LoadResString(errQuickSortFailed)

End Sub

Public Sub QuickSort(vntArray As Variant, _
        Optional ByVal intLBound As Integer, _
        Optional ByVal intUBound As Integer)
    ' Sorts a variant array using Quicksort

    Dim i As Integer
    Dim j As Integer
    Dim vntMid As Variant

    On Error GoTo QuickSortErr

    If IsEmpty(vntArray) Or _
            Not IsArray(vntArray) Then
        Exit Sub
    End If

    ' Set default boundary values for first time through
    If intLBound = 0 And intUBound = 0 Then
        intLBound = LBound(vntArray)
        intUBound = UBound(vntArray)
    End If

    ' BugMessage "Sorting elements " & Str(intLBound) & " and " &
Str(intUBound)

    If intLBound > intUBound Then
        Exit Sub
    End If

    ' Only two elements in this subdivision; exchange if they
    ' are out of order and end recursive calls
    If (intUBound - intLBound) = 1 Then
        If Compare(vntArray(intLBound), vntArray(intUBound)) > 0 Then
            Call Swap(vntArray(intLBound), vntArray(intUBound))
        End If
        Exit Sub
    End If

    ' Set the pivot point
    Set vntMid = vntArray(intUBound)
    i = intLBound
    j = intUBound

    Do
        ' Move in from both sides towards pivot element
        Do While (i < j) And Compare(vntArray(i), vntMid) <= 0
            i = i + 1
        Loop

        Do While (j > i) And Compare(vntArray(j), vntMid) >= 0
            j = j - 1
        Loop
```

```
        If i < j Then
            Call Swap(vntArray(i), vntArray(j))
        End If
    Loop While i < j

    ' Since i has been adjusted, swap element i with element,
    ' intUBound
    Call Swap(vntArray(i), vntArray(intUBound))

    ' Recursively call sort array - pass smaller subdivision
    ' first to conserve stack space
    If (i - intLBound) < (intUBound - 1) Then
        ' Recursively sort with adjusted values for upper and
        ' lower bounds
        Call QuickSort(vntArray, intLBound, i - 1)
        Call QuickSort(vntArray, i + 1, intUBound)
    Else
        Call QuickSort(vntArray, i + 1, intUBound)
        Call QuickSort(vntArray, intLBound, i - 1)
    End If
    Exit Sub

QuickSortErr:
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errQuickSortFailed, mstrModuleName &
"QuickSort", _
        LoadResString(errQuickSortFailed)

End Sub
```

Startup.bas

```
Attribute VB_Name = "Startup"
'  FILE:      Startup.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    This module contains startup and cleanup functions for the
project.
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

Public Const LISTVIEW_BUTTON = 14

Public gstrProjectPath As String

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "Startup."

Private Sub Initialize()

    On Error GoTo InitializeErr

    ReDim gsContCriteria(gintOnFailureAbort To gintOnFailureAsk) As String
    gsContCriteria(gintOnFailureAbort) = "Abort"
    gsContCriteria(gintOnFailureContinue) = "Continue"
    gsContCriteria(gintOnFailureCompleteSiblings) = "Execute sibling steps and
stop"
    gsContCriteria(gintOnFailureAbortSiblings) = "Abort sibling steps and execute
next parent"
    gsContCriteria(gintOnFailureSkipSiblings) = "Skip sibling steps and execute
next parent"
    gsContCriteria(gintOnFailureAsk) = "Ask"

    ReDim gsExecutionStatus(gintDisabled To gintAborted) As String
    gsExecutionStatus(gintDisabled) = "Disabled"
    gsExecutionStatus(gintPending) = "Pending"
    gsExecutionStatus(gintRunning) = "Running"
    gsExecutionStatus(gintComplete) = "Complete"
    gsExecutionStatus(gintFailed) = "Failed"
    gsExecutionStatus(gintAborted) = "Stopped"

#If Not RUN_ONLY Then
    ' Call a procedure to change the style of the toolbar
    ' on the Step Properties form
    Call SetTBar97(frmSteps.tblConstraintCommands)

#End If

    Call InitRunEngine

    Exit Sub

InitializeErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    gstrSource = mstrModuleName & "Initialize"
    Call ShowError(errInitializeFailed)

End Sub
Sub Main()

    On Error GoTo MainErr

    ' Mousepointer should indicate busy
    Call ShowBusy

    ' Display the Splash screen while we carry out some initialization
    frmSplash.Show
    frmSplash.Refresh

    gstrProjectPath = App.Path

    ' Open the database
    If OpenDBFile() = False Then
        Unload frmSplash
        Exit Sub
    End If

#If Not RUN_ONLY Then
    Load frmMain

    ' Enable the Stop Run menu options only when a workspace is
    ' actually running
    Call EnableStop(False)

    ' Clear all application extension menu items
    Call ClearToolsMenu
#End If

    Call Initialize

    ' Mousepointer - ready to accept user input
    Call ShowFree

    ' Unload the Splash screen and display the main form
    Unload frmSplash

#If RUN_ONLY Then
    frmWorkspaceOpen.Caption = gsCaptionRunWsp

    Call ShowWorkspacesInDb(dbsAttTool)
#Else
    frmMain.Show
#End If

    Exit Sub

MainErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    Call ShowFree
```

```
   Call ShowError(errMainFailed)

End Sub
Private Function OpenDBFile() As Boolean
   Dim sDb As String

   On Error GoTo OpenDBFileErr

#If RUN_ONLY Then
   ' Always use the registry setting for the run_only mode
   sDb = DefaultDBFile()
#Else
   ' Check if the user has specified the workspace defn. file to open on the
command line
   ' Else, use the registry setting
   sDb = IIf(StringEmpty(Command), DefaultDBFile(), Command)

   If Len(sDb) > 0 Then
      ' Trim off the enclosing double-quotes if any
      If Mid(sDb, 1, 1) = gstrDQ Then
         If Len(sDb) > 1 Then
            sDb = Mid(sDb, 2)
         Else
            sDb = gstrEmptyString
         End If
      End If
   End If

   If Len(sDb) > 0 Then
      If Mid(sDb, Len(sDb), 1) = gstrDQ Then
         If Len(sDb) > 1 Then
            sDb = Mid(sDb, 1, Len(sDb) - 1)
         Else
            sDb = gstrEmptyString
         End If
      End If
   End If
#End If

   ' Open the database
   OpenDBFile = SMOpenDatabase(sDb)

   Exit Function

OpenDBFileErr:
   Call LogErrors(Errors)
   OpenDBFile = False

End Function
Public Sub Cleanup()

   On Error GoTo CleanupErr

   ' Set the mousepointer to indicate Busy
   Call ShowBusy

#If Not RUN_ONLY Then
   ' Close all open workspaces - will also prompt for unsaved
   ' changes
   Call CloseOpenWorkspaces
#End If

   ' Close all open files
   Call CloseOpenFiles

   ' Reset the mousepointer
   Call ShowFree

   Exit Sub

CleanupErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   Resume Next

End Sub
```

StepCommon.bas

```
Attribute VB_Name = "StepCommon"
'  FILE:      StepCommon.bas
'             Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:   Contains functionality common across StepMaster and
'             SMRunOnly, pertaining to steps
'             Specifically, functions to load iterators records
'             in an array, determine the type of step, etc.
'  Contact:   Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' Used to indicate the source module name when errors
' are raised by this module
Private Const mstrModuleName As String = "StepCommon."

' Step property constants
Private Const mintMinFailureThreshold As Integer = 1
Public Const gintMinSequenceNo As Integer = 1
Public Const gintMinLevel As Integer = 0
Public Function ValidateParallelism(sParallelism As String, lWorkspace As Long,
_
      Optional ParamsInWsp As cArrParameters = Nothing) As String
   ' Returns the degree of parallelism for the step if the user input is valid
   Dim sTemp As String

   On Error GoTo ValidateParallelismErr
   gstrSource = mstrModuleName & "ValidateParallelism"

   sTemp = SubstituteParameters(Trim$(sParallelism), lWorkspace,
WspParameters:=ParamsInWsp)

   If Not IsNumeric(sTemp) Then
      ShowError errInvalidDegParallelism
      On Error GoTo 0
      Err.Raise vbObjectError + errInvalidDegParallelism, gstrSource, _
            LoadResString(errInvalidDegParallelism)
   Else
      If (CInt(sTemp) < gintMinParallelism) Or (CInt(sTemp) >
gintMaxParallelism) Then
         ShowError errInvalidDegParallelism
         On Error GoTo 0
         Err.Raise vbObjectError + errInvalidDegParallelism, gstrSource, _
               LoadResString(errInvalidDegParallelism)
      Else
         ValidateParallelism = Trim$(sParallelism)
      End If
   End If

   Exit Function

ValidateParallelismErr:
   ' Log the error code raised by Visual Basic
   gstrSource = mstrModuleName & "ValidateParallelism"
   If Err.Number = vbObjectError + errSubValuesFailed Then
      ShowError errInvalidDegParallelism
   End If

   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errInvalidDegParallelism, gstrSource, _
      LoadResString(errInvalidDegParallelism)

End Function
```

```
Public Function IsGlobal( _
      Optional ByVal StepClass As cStep = Nothing, _
      Optional ByVal StepRecord As Recordset = Nothing, _
      Optional ByVal StepKey As String = gstrEmptyString, _
      Optional ByVal StepId As Long = 0, _
      Optional StepForm As Form = Nothing) As Boolean

   ' This function contains all the possible checks for whether
   ' a step is global - The check that will be made depends on
   ' the parameter passed in

   Dim cStepRecord As cStep

   If Not StepClass Is Nothing Then
      IsGlobal = StepClass.GlobalFlag
      Exit Function
   End If

   If Not StepRecord Is Nothing Then
      IsGlobal = StepRecord![global_flag]
      Exit Function
   End If

   If Not StringEmpty(StepKey) Then
      IsGlobal = InStr(StepKey, gstrGlobalStepPrefix) > 0
      Exit Function
   End If

   If StepId <> 0 Then
      Set cStepRecord = gcSteps.QueryStep(StepId)
      IsGlobal = cStepRecord.GlobalFlag
      Set cStepRecord = Nothing
      Exit Function
   End If

   If Not StepForm Is Nothing Then
      IsGlobal = (StepForm.lblStepType.Caption = Str(gintGlobalStep))
      Exit Function
   End If

   ' Not a single object was passed in! - raise an error
   On Error GoTo 0
   Err.Raise vbObjectError + errObjectMandatory, _
         mstrModuleName & "IsGlobal", _
         LoadResString(errObjectMandatory)

End Function
Public Function TypeOfStep(Optional ByVal StepClass As cStep = Nothing, _
      Optional ByVal StepRecord As Recordset = Nothing, _
      Optional ByVal StepKey As String = gstrEmptyString, _
      Optional ByVal StepId As Long = 0, _
      Optional StepForm As Form = Nothing) As Integer
   ' Calls functions to determine the type of step
   ' The check that will be made depends on the parameter passed in

   On Error GoTo TypeOfStepErr

   ' Make the check whether a step is global first - both
   ' worker and global steps have the step text or file name
   ' not null - but only the global step will have the global
   ' flag set
   If IsGlobal(StepClass, StepRecord, StepKey, StepId, StepForm) Then
      TypeOfStep = gintGlobalStep
   ElseIf IsManager(StepClass, StepRecord, StepKey, StepId, StepForm) Then
      TypeOfStep = gintManagerStep
   ElseIf IsWorker(StepClass, StepRecord, StepKey, StepId, StepForm) Then
      TypeOfStep = gintWorkerStep
   Else
      On Error GoTo 0
      Err.Raise vbObjectError + errInvalidStep, _
            mstrModuleName & "TypeOfStep", _
            LoadResString(errInvalidStep)
   End If
   Exit Function

TypeOfStepErr:
   ' Log the error code raised by Visual Basic
   Call LogErrors(Errors)
   On Error GoTo 0
   Err.Raise vbObjectError + errTypeOfStepFailed, _
         mstrModuleName & "TypeOfStep", _
         LoadResString(errTypeOfStepFailed)

End Function

Public Function IsStep(intNodeType As Integer) As Boolean
   ' Returns true if the node type corresponds to a global, manager
   ' or worker step
   IsStep = (intNodeType = gintGlobalStep) Or (intNodeType =
gintManagerStep) Or _
      (intNodeType = gintWorkerStep)

End Function

Public Function IsManager(Optional ByVal StepClass As cStep = Nothing, _
      Optional ByVal StepRecord As Recordset = Nothing, _
      Optional ByVal StepKey As String = gstrEmptyString, _
      Optional ByVal StepId As Long = 0, _
      Optional StepForm As Form = Nothing) As Boolean

   ' This function contains all the possible checks for whether
   ' a step is a manager step - The check that will be made depends
   ' on the parameter passed in

   Dim cStepRecord As cStep

   If Not StepClass Is Nothing Then
      IsManager = (StepClass.StepType = gintManagerStep)
      Exit Function
   End If

   If Not StepRecord Is Nothing Then
      IsManager = (IsNull(StepRecord![step_text]) And
IsNull(StepRecord![step_file_name]))
      Exit Function
   End If

   If Not StringEmpty(StepKey) Then
      IsManager = (InStr(StepKey, gstrManagerStepPrefix) > 0)
      Exit Function
   End If

   If StepId <> 0 Then
      Set cStepRecord = gcSteps.QueryStep(StepId)
      IsManager = (cStepRecord.StepType = gintManagerStep)
      Set cStepRecord = Nothing
      Exit Function
   End If

   If Not StepForm Is Nothing Then
      IsManager = (StepForm.lblStepType.Caption = Str(gintManagerStep))
      Exit Function
   End If

   ' Not a single object was passed in! - raise an error
   On Error GoTo 0
   Err.Raise vbObjectError + errObjectMandatory, _
         "Step.IsManager", _
         LoadResString(errObjectMandatory)

End Function
Public Function IsWorker( _
      Optional ByVal StepClass As cStep = Nothing, _
      Optional ByVal StepRecord As Recordset = Nothing, _
      Optional ByVal StepKey As String = gstrEmptyString, _
      Optional ByVal StepId As Long = 0, _
      Optional StepForm As Form = Nothing) As Boolean
```

```
    ' This function contains all the possible checks for whether
    ' a step is a Worker step - The check that will be made depends
    ' on the parameter passed in

    Dim cStepRecord As cStep

    If Not StepClass Is Nothing Then
        IsWorker = (StepClass.StepType = gintWorkerStep)
        Exit Function
    End If

    If Not StepRecord Is Nothing Then
        IsWorker = (Not StepRecord![global_flag] And _
                (Not IsNull(StepRecord![step_text]) Or Not
IsNull(StepRecord![step_file_name])))
        Exit Function
    End If

    If Not StringEmpty(StepKey) Then
        IsWorker = InStr(StepKey, gstrWorkerStepPrefix) > 0
        Exit Function
    End If

    If StepId <> 0 Then
        Set cStepRecord = gcSteps.QueryStep(StepId)
        IsWorker = (cStepRecord.StepType = gintWorkerStep)
        Set cStepRecord = Nothing
        Exit Function
    End If

    If Not StepForm Is Nothing Then
        IsWorker = (StepForm.lblStepType.Caption = Str(gintWorkerStep))
        Exit Function
    End If

    ' Not a single object was passed in! - raise an error
    On Error GoTo 0
    Err.Raise vbObjectError + errObjectMandatory, _
            "Step.IsWorker", _
            LoadResString(errObjectMandatory)

End Function
Public Function GetStepNodeText(ByVal cStepNode As cStep) As String

    On Error GoTo GetStepNodeTextErr

    ' Returns the string that will be displayed as the text
    ' in the tree view node to the user
    If StringEmpty(cStepNode.StepLabel) Then

        If StringEmpty(cStepNode.StepTextFile) Then

            If StringEmpty(cStepNode.StepText) Then
                ' This should never happen
                On Error GoTo 0
                Err.Raise vbObjectError + errStepLabelTextAndFileNull, _
                    gstrSource, _
                    LoadResString(errStepLabelTextAndFileNull)
            Else
                GetStepNodeText = cStepNode.StepText
            End If
        Else
            GetStepNodeText = cStepNode.StepTextFile
        End If
    Else
        GetStepNodeText = cStepNode.StepLabel
    End If

    Exit Function

GetStepNodeTextErr:
    ' Log the error code raised by Visual Basic
    Call LogErrors(Errors)
    On Error GoTo 0
    Err.Raise vbObjectError + errGetStepNodeTextFailed, _
            gstrSource, _
            LoadResString(errGetStepNodeTextFailed)

End Function

Public Function LoadRecordsetInStepsArray(rstSteps As Recordset, _
    cStepCol As cArrSteps) As Boolean

    Dim cNewStep As cStep
    Dim cNewGlobal As cGlobalStep
    Dim cNewManager As cManager
    Dim cNewWorker As cWorker

    On Error GoTo LoadRecordsetInStepsArrayErr

    If rstSteps.RecordCount = 0 Then
        Exit Function
    End If

    rstSteps.MoveFirst
    While Not rstSteps.EOF
        ' For fields that should not be null, a procedure is first
        ' called to raise an error if the field is null

        Set cNewStep = New cStep

        cNewStep.StepType = TypeOfStep(StepRecord:=rstSteps)

        If cNewStep.StepType = gintGlobalStep Then
            Set cNewGlobal = New cGlobalStep
            Set cNewStep = cNewGlobal
        ElseIf cNewStep.StepType = gintManagerStep Then
            Set cNewManager = New cManager
            Set cNewStep = cNewManager
        Else
            Set cNewWorker = New cWorker
            Set cNewStep = cNewWorker
        End If

        ' Initialize the global flag first, since subsequent
        ' validations might depend on whether the step is global
        cNewStep.GlobalFlag = CBool(ErrorOnNullField(rstSteps, "global_flag"))

        ' Initialize step values
        cNewStep.StepId = CLng(ErrorOnNullField(rstSteps, "step_id"))
        cNewStep.VersionNo = CStr(ErrorOnNullField(rstSteps, "version_no"))

        cNewStep.StepLabel = CheckForNullField(rstSteps, "step_label")
        cNewStep.StepTextFile = CheckForNullField(rstSteps, "step_file_name")
        cNewStep.StepText = CheckForNullField(rstSteps, "step_text")
        cNewStep.StartDir = CheckForNullField(rstSteps, "start_directory")

        cNewStep.WorkspaceId = CLng(ErrorOnNullField(rstSteps,
FLD_ID_WORKSPACE))
        cNewStep.ParentStepId = CLng(ErrorOnNullField(rstSteps,
"parent_step_id"))
        cNewStep.ParentVersionNo = CStr(ErrorOnNullField(rstSteps,
"parent_version_no"))

        cNewStep.SequenceNo = CInt(ErrorOnNullField(rstSteps, "sequence_no"))
        cNewStep.StepLevel = CInt(ErrorOnNullField(rstSteps, "step_level"))
        cNewStep.EnabledFlag = CBool(ErrorOnNullField(rstSteps,
"enabled_flag"))

        ' Initialize the execution details for the step
        cNewStep.DegreeParallelism = CheckForNullField(rstSteps,
"degree_parallelism")
        cNewStep.ExecutionMechanism = CInt(ErrorOnNullField(rstSteps,
"execution_mechanism"))
        cNewStep.FailureDetails = CheckForNullField(rstSteps, "failure_details")
        cNewStep.ContinuationCriteria = CInt(ErrorOnNullField(rstSteps,
"continuation_criteria"))
```

```
        ' Initialize the output file locations for the step
        cNewStep.OutputFile = CheckForNullField(rstSteps, "output_file_name")
        ' cNewStep.LogFile = CheckForNullField(rstSteps, "log_file_name")
        cNewStep.ErrorFile = CheckForNullField(rstSteps, "error_file_name")

        ' Initialize the iterator name for the step, if any
        cNewStep.IteratorName = CheckForNullField(rstSteps, "iterator_name")

        ' Add this record to the array of steps
        cStepCol.Load cNewStep

        Set cNewStep = Nothing
        rstSteps.MoveNext
    Wend

    Exit Function

LoadRecordsetInStepsArrayErr:

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

End Function
```

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
```

TPCHReports.bas

```
Attribute VB_Name = "TPCHReports"
'  FILE:      TPCHReports.bas
'             Reporting Program for Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:   This is the main module that contains the report generation
'             functions for TPCHMetrics
'  Contact:   Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

' DAO workspace, database and recordset objects
Private mwrkJet As DAO.Workspace
Private mDB As DAO.Database
Private mrecDtls As DAO.Recordset

' Constants for Power and Throughput runs
' Must correspond to index for checkboxes, chkRun on frmTPCHMetrics
Public Enum eRunType
   Power = 1
   Throughput = 2
End Enum

' Range of streams for which the Power/Throughput run reports can be generated
Private Const miStreamIdMin As Integer = 0 ' Corresponds to PowerRun
Private Const miStreamIdMax As Integer = 10

' Power Run corresponds to Stream 0
Private Const miStreamPowerRun As Integer = miStreamIdMin

' Range of Power/Throughput run queries
Private Const miQueryIdMin As Integer = 1
Private Const miQueryIdMax As Integer = 24 ' Includes RF functions

' Constant for Query 15 since it is split into view creation and execution
' in the non-standard version of the kit
Private Const miQuery15 As Integer = 15

' Constants for refresh functions, RF1 and RF2
Private Const miQueryIdRF1 As Integer = 23
Private Const miQueryIdRF2 As Integer = 24

' Number of refresh functions
Private Const miRefreshFuncs As Integer = 2

' Constant for the first iterator value for the refresh functions
Private Const msFirstRFIterator As String = "1"

' Upper bound for Scale Factor (& number of streams) array
Private Const miSFUBound As Integer = 8

' Array of valid scale factors
Private marrSF(miSFUBound) As Single
' Array of minimum number of throughput streams for each scale factor
Private marrStreams(miSFUBound) As Integer

' Default scale factor array location
Private Const miDefaultSF As Integer = 0

' Default number of insert segments per update set
Private Const miDefInsSegmentsPerUpdateSet As Integer = 16
' Default number of delete segments per update set
Private Const miDefDelSegmentsPerUpdateSet As Integer = 16

#If Not TPCH_KIT Then
   ' Default value for first update set for throughput run
   Private Const miDefUpdateSetFirst As Integer = 2
#End If

' Module-level variables for all report inputs
Private msDbFile As String      ' Workspace definition file name
Private msOutput As String      ' Report file name
Private mbRunType(eRunType.Power To eRunType.Throughput) As Boolean
                        ' Array with Yes/No values for type of report
                        '  being generated
Private miStreamIdLo As Integer ' Will be initialized to 0 or 1, depending on
whether the
                        ' PowerRun Metric is being generated or not
Private miStreamIdHi As Integer ' Will be initialized to number of throughput
                        ' streams or 0, depending on whether the
                        ' ThroughputRun Metric is being generated or not
Private mlRunId As Long         ' Run identifier
Private mfSF As Single          ' Scale factor for the run
Private miInsSegmentsPerUpdateSet As Integer    ' Number of insert segments per
update set
Private miDelSegmentsPerUpdateSet As Integer    ' Number of delete segments
per update set

#If Not TPCH_KIT Then
   Private miUpdateSetFirst As Integer         ' First update set for throughput run
#End If

' Constants for each of the dimensions (stream and query ) in the
arQuerySequence array
Private Const miDimensionStream As Integer = 1
Private Const miDimensionQuery As Integer = 2

' Array containing the sequence in which queries are executed, excluding the
refresh functions
Private arQuerySequence(miStreamIdMin To miStreamIdMax, miQueryIdMin To
miQueryIdMax - miRefreshFuncs) As Integer

' Stucture containing step execution statistics
Private Type StepExecutionDtls
   StartTime As Currency
   EndTime As Currency
   StartTimeNextStep As Currency
   ElapsedTime As Double
   ElapsedTpch As Double
   sStartTm As String
   sEndTm As String
   sStartNextTm As String
End Type

' Array containing step execution statistics for each query in each stream
Private arExecutionDtls(miStreamIdMin To miStreamIdMax, miQueryIdMin To
miQueryIdMax) As StepExecutionDtls

' Constants used to format dates and timestamps
Private Const msDtSeparator As String = "-"
Private Const msTmSeparator As String = ":"
Private Const msMsSeparator As String = "."
Private Const msBlank As String = " "
Private Const msFmtMsPadWZeroes As String = "000"
Private Const msFmtYrPadWZeroes As String = "0000"

' Constants used to format stream and query identifiers
Private Const msFmtNumberWOZeroes As String = "##"
Private Const msFmtNumberPadWZeroes As String = "00"

' Constant used to format fractions
Private Const miNumDecimalPlaces As Integer = 1

' Constants used by the metrics calculation functions
Private Const mfMinElapsed As Single = 0.05
Private Const mfMinElapsedRound As Single = 0.1

' String constants for labels in workspace. Edit these labels if the workspace is
modified.
' Two versions of the workspace currently exist - the one in the tpch kit and the
```

```
' one that Larry and Grace use
' PowerRun labels
#If TPCH_KIT Then
   Private Const msStartStepLabelPowerRF1 As String = "Power - Execute RF1"
   Private Const msStartStepLabelPowerRF2 As String = "Power - Execute RF2"
   Private Const msStartStepLabelPowerOther As String = "Power - Execute
Query "

   Private Const msEndStepLabelPowerRF1 As String = "Power - Execute RF1"
   Private Const msEndStepLabelPowerRF2 As String = "Power - Execute RF2"
#Else
   Private Const msStartStepLabelPowerRF1 As String = "Execute RF1 in
Parallel"
   Private Const msStartStepLabelPowerRF2 As String = "Execute Query RF2
Thread"
   Private Const msStartStepLabelPower15 As String = "Create REVENUE
View"
   Private Const msStartStepLabelPowerOther As String = "Execute Query "

   Private Const msEndStepLabelPowerRF1 As String = "Execute RF1 in
Parallel"
   Private Const msEndStepLabelPowerRF2 As String = "Execute Query RF2
Thread"
   Private Const msEndStepLabelPower15 As String = "Execute using
REVENUE"
#End If

' ThroughputRun labels
#If TPCH_KIT Then
   Private Const msStartStepLabelThroughputRF1First As String = "Throughput -
Semaphore Loop for RF Delay"
   Private Const msEndStepLabelThroughputRF2Rest As String = "Throughput -
Stream - Increment Update Set"

   Private Const msLabelThroughputRF As String = "Sequential Refresh Stream
Execution"
   Private Const msLblThroughputRF12Mgr As String = "Throughput - Refresh
Streams"
   Private Const msLabelThroughputRF1 As String = "Throughput - RF1 -
Stream%STREAM_NUM%"
   Private Const msLabelThroughputRF2 As String = "Throughput - RF2 -
Stream%STREAM_NUM%"
   Private Const msLblThroughputRF1 As String = "Throughput - Execute
Stream%STREAM_NUM% RF1"
   Private Const msLblThroughputRF2 As String = "Throughput - Execute
Stream%STREAM_NUM% RF2"

   Private Const msStepLabelThroughputOther As String = "Throughput - Query
Stream "
#Else
   Private Const msStartStepLabelThroughputRF1 As String = "Wait for
Throughput Run"
   Private Const msStartStepLabelThroughputRF2 As String = "Throughput RF2"
   Private Const msStartStepLabelThroughputOther As String = "Execute Query
for Stream "

   Private Const msEndStepLabelThroughputRF1 As String = "Throughput RF1"

   Private Const msLabelThroughputRF As String = "Refresh Streams"
   Private Const msLabelThroughputRFWithIt As String = "Refresh Streams with
Iterators"
   Private Const msLabelThroughputRF1 As String = "RF1"
   Private Const msLabelThroughputRF2 As String = "RF2"
   Private Const msLabelThroughputRF12 As String = "RF1 post Throughput
Queries"
#End If

Private Sub CalcElapsedTmAndPrint()
   ' This is the main function that computes the elapsed time for each
   ' step, the Qpp for the Power and Throughput runs and the composite
   ' QppH

   Dim iStreamId As Integer
   Dim iQueryId As Integer
   Dim lMinElapsed As Double
   Dim lMaxElapsed As Double
   Dim fSigma As Double
   Dim fQppPower As Double
   Dim fQppThroughput As Double
   Dim fQppH As Double
   Dim cStart As Currency
   Dim cEnd As Currency
   Dim fElapsedThroughput As Double
   Dim ftemp As Double
   Dim sStartTm As String
   Dim sEndTm As String

   ' Calculate the timing interval for each query in each stream
   For iStreamId = miStreamIdLo To miStreamIdHi

      For iQueryId = miQueryIdMin To miQueryIdMax Step 1

         ' Calculate the elapsed time for a query as the difference
         ' between the end time and the start time for a step
         arExecutionDtls(iStreamId, iQueryId).ElapsedTime = _
               (arExecutionDtls(iStreamId, iQueryId).EndTime - _
                  arExecutionDtls(iStreamId, iQueryId).StartTime) * 10

         ' If the timing interval for a query is less than 0.05 seconds,
         ' it must be rounded up to 0.1 second to avoid 0 values
         ' Refer to Section 5.3.7.5 of the TPC-H Spec Rev. 1.1.0
         If (arExecutionDtls(iStreamId, iQueryId).ElapsedTime) < mfMinElapsed
Then
            arExecutionDtls(iStreamId, iQueryId).ElapsedTime =
mfMinElapsedRound
         Else
            arExecutionDtls(iStreamId, iQueryId).ElapsedTime = _
               Round(arExecutionDtls(iStreamId, iQueryId).ElapsedTime,
miNumDecimalPlaces)
         End If

         ' Calculate the timing interval for the TPC-H Metric as the difference
         ' between the start time of the next step and the start time for a step
         arExecutionDtls(iStreamId, iQueryId).ElapsedTpch = _
               (arExecutionDtls(iStreamId, iQueryId).StartTimeNextStep - _
                  arExecutionDtls(iStreamId, iQueryId).StartTime) * 10

         ' If the timing interval for a query is less than 0.05 seconds,
         ' it must be rounded up to 0.1 second to avoid 0 values
         ' Refer to Section 5.3.7.5 of the TPC-H Spec Rev. 1.1.0
         If (arExecutionDtls(iStreamId, iQueryId).ElapsedTpch) < mfMinElapsed
Then
            arExecutionDtls(iStreamId, iQueryId).ElapsedTpch =
mfMinElapsedRound
         Else
            arExecutionDtls(iStreamId, iQueryId).ElapsedTpch = _
               Round(arExecutionDtls(iStreamId, iQueryId).ElapsedTpch,
miNumDecimalPlaces)
         End If
      Next iQueryId

   Next iStreamId

   If mbRunType(eRunType.Power) Then
      lMinElapsed = arExecutionDtls(miStreamPowerRun,
miQueryIdMin).ElapsedTpch
      lMaxElapsed = 0

      ' Determine minimum and maximum query timing intervals for the
PowerRun
      For iQueryId = miQueryIdMin To miQueryIdMax - miRefreshFuncs Step 1

         If arExecutionDtls(miStreamPowerRun, iQueryId).ElapsedTpch >
lMaxElapsed Then
            lMaxElapsed = arExecutionDtls(miStreamPowerRun,
iQueryId).ElapsedTpch
         End If
```

```
        If arExecutionDtls(miStreamPowerRun, iQueryId).ElapsedTpch <
lMinElapsed Then
            lMinElapsed = arExecutionDtls(miStreamPowerRun,
iQueryId).ElapsedTpch
        End If

      Next iQueryId

      ' If the ratio between the longest query timing interval and the shortest query
      ' timing interval in the PowerRun is greater than 1000, then increase all query
      ' timing intervals which are smaller than max[QI(i,0)]/1000 to
max[QI(i,0)]/1000
      ' Refer to Section 5.4.1.4 of the TPC-H Spec Rev. 1.1.0
      If lMaxElapsed / lMinElapsed > 1000 Then
        lMinElapsed = lMaxElapsed / 1000

        For iQueryId = miQueryIdMin To miQueryIdMax Step 1

            If arExecutionDtls(miStreamPowerRun, iQueryId).ElapsedTpch >
lMaxElapsed Then
                lMaxElapsed = arExecutionDtls(miStreamPowerRun,
iQueryId).ElapsedTpch
            End If

            If arExecutionDtls(miStreamPowerRun, iQueryId).ElapsedTpch <
(lMaxElapsed / 1000) Then
                arExecutionDtls(miStreamPowerRun, iQueryId).ElapsedTpch =
lMinElapsed
            End If

        Next iQueryId
      End If

      ' Determine the Power Metric
      fSigma = 0
      For iQueryId = miQueryIdMin To miQueryIdMax Step 1
        fSigma = fSigma + Log(arExecutionDtls(miStreamPowerRun,
iQueryId).ElapsedTpch)
      Next iQueryId
      fSigma = fSigma * (-1 / 24)
      fQppPower = Round(Exp(fSigma) * 3600 * mfSF, miNumDecimalPlaces)
    End If

    If mbRunType(eRunType.Throughput) Then
      ' Determine the elapsed time for the throughput run
      cStart = arExecutionDtls(miStreamPowerRun + 1, miQueryIdMin).StartTime
      cEnd = 0

      For iStreamId = miStreamPowerRun + 1 To miStreamIdHi

        For iQueryId = miQueryIdMin To miQueryIdMax Step 1

            If arExecutionDtls(iStreamId, iQueryId).StartTime < cStart Then
                cStart = arExecutionDtls(iStreamId, iQueryId).StartTime
            End If

            If arExecutionDtls(iStreamId, iQueryId).EndTime > cEnd Then
                cEnd = arExecutionDtls(iStreamId, iQueryId).EndTime
            End If

        Next iQueryId

      Next iStreamId

      sStartTm = JulianToString(cStart)
      sEndTm = JulianToString(cEnd)

      fElapsedThroughput = Round((cEnd - cStart) * 10, miNumDecimalPlaces)
      fSigma = 3600 / fElapsedThroughput
      fQppThroughput = Round(fSigma * miStreamIdHi * 22 * mfSF,
miNumDecimalPlaces)
    End If

    ' Compute the composite QppH value
    fQppH = Round(Sqr(fQppPower * fQppThroughput), miNumDecimalPlaces)

    Call PrintStatistics(fQppPower, fElapsedThroughput, fQppThroughput, fQppH,
_
        sStartTm, sEndTm)

End Sub
Private Sub CloseDbAndWsp()
    ' Close the database structures

    On Error Resume Next

    If Not mDB Is Nothing Then
        mDB.Close
        Set mDB = Nothing
    End If

    If Not mwrkJet Is Nothing Then
        mwrkJet.Close
        Set mwrkJet = Nothing
    End If

End Sub

Public Sub NumStreamsEnabled(Index As Integer)
    ' Enables the number of throughput streams only if the throughput
    ' metric report is being generated

    If Index = eRunType.Throughput Then
        frmTPCHMetrics.cboStreams.Enabled = _
            (frmTPCHMetrics.chkRun(eRunType.Throughput).Value =
vbChecked)
#If Not TPCH_KIT Then
        ' Enables the first throughput update set only if the throughput
        ' metric report is being generated
        frmTPCHMetrics.txtUpdateSetFirst.Enabled = _
            (frmTPCHMetrics.chkRun(eRunType.Throughput).Value =
vbChecked)
#End If
    End If

End Sub

Private Function ParentInstId(iStreamId As Integer, iQueryId As Integer) As
Long
    ' Determines the parent instance identifier for the refresh functions
    ' Uses the refresh function label that marks the start of execution for
    ' the refresh function
    ' E.g. TPC-H Kit, Stream1, RF1: the start time for the refresh function
    ' is the start time for the step, 'Throughput - Semaphore Loop for RF Delay'
    ' but the end time for the refresh function is the end time for the step,
    ' 'Throughput - Execute Stream1 RF1'
    ' Similarly, in the HP version of the TPC-H Kit, RF1: the start time
    ' for the refresh function is the start time for the step,
    ' 'Wait for Throughput Run' but the end time for the refresh function
    ' is the end time for the step, 'Throughput RF1'

    Dim sLabel As String
    Dim sSql As String
    Dim sIt As String
    Dim lInstId As Long

    ParentInstId = 0

    ' Start at the top level step and work down to the parent of the refresh
    ' function. Initialize the step label and iterator values at every
    ' step, build a query and determine the instance identifier

    sLabel = msLabelThroughputRF
    sSql = "step_label = " & MakeStringFieldValid(sLabel)
    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
        MsgBox "Step " & "'" & sLabel & "'" & " not found! " & _
            "The Throughput Run was not completely executed!"
```

```
        Exit Function
    End If

    lInstId = mrecDtls.Fields("instance_id").Value

#If TPCH_KIT Then
    If (iStreamId = miStreamPowerRun + 1) And iQueryId = miQueryIdRF1 Then
        ParentInstId = lInstId
        Exit Function
    End If

    sLabel = msLblThroughputRF12Mgr
    sIt = CStr(iStreamId)
#Else
    sLabel = msLabelThroughputRFWithIt
    ' Stream 1 in Throughput run has iterator value first update set,
    ' Stream 2 in Throughput run has iterator value first update set + 1, ... and so on
    sIt = Format(iStreamId + miUpdateSetFirst - 1, msFmtNumberWOZeroes)
#End If

    sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
           " and parent_instance_id = " & CStr(lInstId) & _
           " and iterator_value = " & MakeStringFieldValid(sIt)
    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
        MsgBox "Step " & """ & sLabel & """ & _
               " not found for iterator " & """ & sIt & """ & ". " & _
               "The Throughput Run was not completely executed!"
        Exit Function
    End If

    lInstId = mrecDtls.Fields("instance_id").Value

    sLabel = IIf(iQueryId = miQueryIdRF1, msLabelThroughputRF1,
msLabelThroughputRF2)

    sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
           " and parent_instance_id = " & CStr(lInstId)

    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
        MsgBox "Step " & """ & sLabel & """ & _
               " not found for iterator " & """ & sIt & """ & ". " & _
               "The Throughput Run was not completely executed!"
        Exit Function
    End If

    ParentInstId = mrecDtls.Fields("instance_id").Value

End Function
Private Function ParentInstIdEnd(iStreamId As Integer, iQueryId As Integer) As
Long
    ' Determines the parent instance identifier for the refresh functions
    ' Uses the refresh function label that marks the end of execution for
    ' the refresh function
    ' E.g. TPC-H Kit, Stream1, RF1: the start time for the refresh function
    ' is the start time for the step, 'Throughput - Semaphore Loop for RF Delay'
    ' but the end time for the refresh function is the end time for the step,
    ' 'Throughput - Execute Stream1 RF1'
    ' Similarly, in the HP version of the TPC-H Kit, RF1: the start time
    ' for the refresh function is the start time for the step,
    ' 'Wait for Throughput Run' but the end time for the refresh function
    ' is the end time for the step, 'Throughput RF1'

    Dim sLabel As String
    Dim sSql As String
    Dim sIt As String
    Dim lInstId As Long

    ' Start at the top level step and work down to the parent of the refresh
    ' function. Initialize the step label and iterator values at every
    ' step, build a query and determine the instance identifier

    ParentInstIdEnd = 0

#If TPCH_KIT Then
    sLabel = msLabelThroughputRF
    sSql = "step_label = " & MakeStringFieldValid(sLabel)
    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
        MsgBox "Step " & """ & sLabel & """ & " not found! " & _
               "The Throughput Run was not completely executed!"
        Exit Function
    End If

    lInstId = mrecDtls.Fields("instance_id").Value

    sLabel = msLblThroughputRF12Mgr
    sIt = CStr(iStreamId)
    sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
           " and parent_instance_id = " & CStr(lInstId) & _
           " and iterator_value = " & MakeStringFieldValid(sIt)
    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
        MsgBox "Step " & """ & sLabel & """ & _
               " not found for iterator " & """ & sIt & """ & ". " & _
               "The Throughput Run was not completely executed!"
        Exit Function
    End If

    lInstId = mrecDtls.Fields("instance_id").Value

    If iQueryId = miQueryIdRF2 And iStreamId <> miStreamIdHi Then
        ParentInstIdEnd = lInstId
        Exit Function
    End If

    sLabel = IIf(iQueryId = miQueryIdRF1, msLabelThroughputRF1,
msLabelThroughputRF2)
    sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
           " and parent_instance_id = " & CStr(lInstId)
    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
        MsgBox "Step " & """ & sLabel & """ & " not found! " & _
               "The Throughput Run was not completely executed!"
        Exit Function
    End If

    ParentInstIdEnd = mrecDtls.Fields("instance_id").Value
#Else
    sLabel = msLabelThroughputRF
    sSql = "step_label = " & MakeStringFieldValid(sLabel)

    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
        MsgBox "Step " & """ & sLabel & """ & _
               " not found for iterator " & """ & sIt & """ & ". " & _
               "The Throughput Run was not completely executed!"
        Exit Function
    End If

    lInstId = mrecDtls.Fields("instance_id").Value

    ' Stream 1 in Throughput run has iterator value first update set,
    ' Stream 2 in Throughput run has iterator value first update set + 1, ... and so on
    sIt = Format(iStreamId + miUpdateSetFirst - 1, msFmtNumberWOZeroes)
    sLabel = msLabelThroughputRFWithIt
    sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
           " and parent_instance_id = " & CStr(lInstId) & _
           " and iterator_value = " & MakeStringFieldValid(sIt)

    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
        MsgBox "Step " & """ & sLabel & """ & _
               " not found for iterator " & """ & sIt & """ & ". " & _
               "The Throughput Run was not completely executed!"
        Exit Function
    End If
```

```
    lInstId = mrecDtls.Fields("instance_id").Value
    sLabel = IIf(iQueryId = miQueryIdRF1, msLabelThroughputRF1,
msLabelThroughputRF2)

    sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
           " and parent_instance_id = " & CStr(lInstId)

    mrecDtls.FindFirst sSql
    If mrecDtls.NoMatch = True Then
       MsgBox "Step " & "'" & sLabel & "'" & _
              " not found for iterator " & "'" & sIt & "'" & ". " & _
              "The Throughput Run was not completely executed!"
       Exit Function
    End If

    lInstId = mrecDtls.Fields("instance_id").Value

    If iQueryId = miQueryIdRF1 Then
       sLabel = msLabelThroughputRF12

       sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
              " and parent_instance_id = " & CStr(lInstId)

       mrecDtls.FindFirst sSql
       If mrecDtls.NoMatch = True Then
          MsgBox "Step " & "'" & sLabel & "'" & _
                 " not found for iterator " & "'" & sIt & "'" & ". " & _
                 "The Throughput Run was not completely executed!"
          Exit Function
       End If

       lInstId = mrecDtls.Fields("instance_id").Value
    End If

    ParentInstIdEnd = lInstId

#End If

End Function
Public Sub Initialize()

    ' Initialize an array of valid scale factors
    marrSF(0) = 0.1
    marrSF(1) = 1
    marrSF(2) = 10
    marrSF(3) = 30
    marrSF(4) = 100
    marrSF(5) = 300
    marrSF(6) = 1000
    marrSF(7) = 3000
    marrSF(8) = 10000

    ' and the number of streams for each scale factor
    marrStreams(0) = 2
    marrStreams(1) = 2
    marrStreams(2) = 3
    marrStreams(3) = 4
    marrStreams(4) = 5
    marrStreams(5) = 6
    marrStreams(6) = 7
    marrStreams(7) = 8
    marrStreams(8) = 9

    ' Load the valid scale factors and number of streams for each
    ' scale factor in the combo boxes on the form
    Call LoadSFAndStreams(frmTPCHMetrics.cboSF,
frmTPCHMetrics.cboStreams)

    ' Initialize the number of insert and delete segments per update
    ' set to default values
    frmTPCHMetrics.txtInsertSegments.Text = miDefInsSegmentsPerUpdateSet
    frmTPCHMetrics.txtDeleteSegments.Text = miDefDelSegmentsPerUpdateSet
#If TPCH_KIT Then
    frmTPCHMetrics.lblFirstUpdateSet.Enabled = False
    frmTPCHMetrics.txtUpdateSetFirst.Enabled = False
#Else
    frmTPCHMetrics.lblFirstUpdateSet.Enabled = True
    frmTPCHMetrics.txtUpdateSetFirst.Enabled = True
    ' Initialize the first update set for the throughput run
    frmTPCHMetrics.txtUpdateSetFirst.Text = miDefUpdateSetFirst
#End If

    ' Initialize the sequence in which queries are executed for PowerRun
    arQuerySequence(miStreamPowerRun, miQueryIdMin) = 14
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 1) = 2
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 2) = 9
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 3) = 20
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 4) = 6
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 5) = 17
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 6) = 18
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 7) = 8
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 8) = 21
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 9) = 13
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 10) = 3
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 11) = 22
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 12) = 16
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 13) = 4
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 14) = 11
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 15) = 15
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 16) = 1
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 17) = 10
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 18) = 19
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 19) = 5
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 20) = 7
    arQuerySequence(miStreamPowerRun, miQueryIdMin + 21) = 12

    ' Initialize the sequence in which throughput run queries are executed
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin) = 21
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 1) = 3
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 2) = 18
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 3) = 5
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 4) = 11
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 5) = 7
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 6) = 6
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 7) = 20
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 8) = 17
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 9) = 12
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 10) = 16
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 11) = 15
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 12) = 13
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 13) = 10
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 14) = 2
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 15) = 8
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 16) = 14
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 17) = 19
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 18) = 9
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 19) = 22
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 20) = 1
    arQuerySequence(miStreamPowerRun + 1, miQueryIdMin + 21) = 4

    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin) = 6
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 1) = 17
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 2) = 14
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 3) = 16
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 4) = 19
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 5) = 10
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 6) = 9
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 7) = 2
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 8) = 15
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 9) = 8
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 10) = 5
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 11) = 22
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 12) = 12
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 13) = 7
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 14) = 13
    arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 15) = 18
```

arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 16) = 1
arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 17) = 4
arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 18) = 20
arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 19) = 3
arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 20) = 11
arQuerySequence(miStreamPowerRun + 2, miQueryIdMin + 21) = 21

arQuerySequence(miStreamPowerRun + 3, miQueryIdMin) = 8
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 1) = 5
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 2) = 4
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 3) = 6
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 4) = 17
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 5) = 7
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 6) = 1
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 7) = 18
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 8) = 22
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 9) = 14
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 10) = 9
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 11) = 10
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 12) = 15
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 13) = 11
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 14) = 20
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 15) = 2
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 16) = 21
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 17) = 19
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 18) = 13
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 19) = 16
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 20) = 12
arQuerySequence(miStreamPowerRun + 3, miQueryIdMin + 21) = 3

arQuerySequence(miStreamPowerRun + 4, miQueryIdMin) = 5
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 1) = 21
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 2) = 14
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 3) = 19
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 4) = 15
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 5) = 17
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 6) = 12
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 7) = 6
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 8) = 4
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 9) = 9
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 10) = 8
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 11) = 16
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 12) = 11
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 13) = 2
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 14) = 10
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 15) = 18
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 16) = 1
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 17) = 13
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 18) = 7
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 19) = 22
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 20) = 3
arQuerySequence(miStreamPowerRun + 4, miQueryIdMin + 21) = 20

arQuerySequence(miStreamPowerRun + 5, miQueryIdMin) = 21
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 1) = 15
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 2) = 4
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 3) = 6
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 4) = 7
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 5) = 16
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 6) = 19
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 7) = 18
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 8) = 14
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 9) = 22
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 10) = 11
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 11) = 13
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 12) = 3
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 13) = 1
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 14) = 2
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 15) = 5
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 16) = 8
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 17) = 20
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 18) = 12
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 19) = 17
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 20) = 10
arQuerySequence(miStreamPowerRun + 5, miQueryIdMin + 21) = 9

arQuerySequence(miStreamPowerRun + 6, miQueryIdMin) = 10
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 1) = 3
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 2) = 15
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 3) = 13
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 4) = 6
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 5) = 8
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 6) = 9
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 7) = 7
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 8) = 4
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 9) = 11
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 10) = 22
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 11) = 18
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 12) = 12
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 13) = 1
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 14) = 5
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 15) = 16
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 16) = 2
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 17) = 14
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 18) = 19
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 19) = 20
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 20) = 17
arQuerySequence(miStreamPowerRun + 6, miQueryIdMin + 21) = 21

arQuerySequence(miStreamPowerRun + 7, miQueryIdMin) = 18
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 1) = 8
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 2) = 20
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 3) = 21
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 4) = 2
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 5) = 4
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 6) = 22
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 7) = 17
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 8) = 1
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 9) = 11
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 10) = 9
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 11) = 19
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 12) = 3
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 13) = 13
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 14) = 5
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 15) = 7
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 16) = 10
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 17) = 16
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 18) = 6
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 19) = 14
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 20) = 15
arQuerySequence(miStreamPowerRun + 7, miQueryIdMin + 21) = 12

arQuerySequence(miStreamPowerRun + 8, miQueryIdMin) = 19
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 1) = 1
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 2) = 15
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 3) = 17
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 4) = 5
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 5) = 8
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 6) = 9
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 7) = 12
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 8) = 14
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 9) = 7
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 10) = 4
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 11) = 3
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 12) = 20
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 13) = 16
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 14) = 6
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 15) = 22
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 16) = 10
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 17) = 13
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 18) = 2
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 19) = 21
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 20) = 18
arQuerySequence(miStreamPowerRun + 8, miQueryIdMin + 21) = 11

arQuerySequence(miStreamPowerRun + 9, miQueryIdMin) = 8
arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 1) = 13
arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 2) = 2

```
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 3) = 20
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 4) = 17
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 5) = 3
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 6) = 6
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 7) = 21
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 8) = 18
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 9) = 11
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 10) = 19
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 11) = 10
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 12) = 15
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 13) = 4
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 14) = 22
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 15) = 1
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 16) = 7
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 17) = 12
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 18) = 9
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 19) = 14
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 20) = 5
    arQuerySequence(miStreamPowerRun + 9, miQueryIdMin + 21) = 16

    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin) = 6
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 1) = 15
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 2) = 18
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 3) = 17
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 4) = 12
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 5) = 1
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 6) = 7
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 7) = 2
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 8) = 22
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 9) = 13
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 10) = 21
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 11) = 10
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 12) = 14
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 13) = 9
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 14) = 3
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 15) = 16
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 16) = 20
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 17) = 19
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 18) = 11
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 19) = 4
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 20) = 8
    arQuerySequence(miStreamPowerRun + 10, miQueryIdMin + 21) = 5

End Sub

Private Sub QueryRunDetails()
    ' Queries step execution records from the database

    Dim strSql As String

    On Error GoTo QueryRunDetailsErr

    Screen.MousePointer = vbHourglass

    Set mwrkJet = CreateWorkspace("tpch", "admin", gsEmptyString, dbUseJet)

    ' Open database for exclusive access
    Set mDB = mwrkJet.OpenDatabase(msDbFile, Options:=True)

    strSql = "select a.step_id, a.version_no, a.command, " & _
            " a.instance_id, a.parent_instance_id, a.iterator_value, " & _
            " a.start_time, a.end_time, " & _
            " b.step_label, b.step_file_name, b.step_text, " & _
            " b.step_level, b.sequence_no, " & _
            " b.iterator_name " & _
            " from run_step_details a, att_steps b " & _
            " where a.step_id = b.step_id " & _
            " and a.version_no = b.version_no " & _
            " and a.run_id = " & CStr(mlRunId) & _
            " order by b.step_level, b.sequence_no, a.instance_id"

    Set mrecDtls = mDB.OpenRecordset(strSql, dbOpenSnapshot)
    If mrecDtls.RecordCount = 0 Then
        Screen.MousePointer = vbDefault
        MsgBox "There are no records in the database for the run identifier." & _
                " Unable to generate reports."
    Else
        Call CollectStats
    End If

    mrecDtls.Close
    CloseDbAndWsp

    Screen.MousePointer = vbDefault

    Exit Sub

QueryRunDetailsErr:

    Screen.MousePointer = vbDefault
    DisplayErrors

    CloseDbAndWsp

End Sub
Public Sub GenerateSelected()
    ' Called when the user clicks on Generate on the TPCHMetrics form.
    ' Validates all the user inputs and calls a function to generate
    ' the reports if inputs are valid.
    ' Initializes module-level variables to the values input by the user
    Dim sTemp As String

    On Error GoTo GenerateSelectedErr

    ' Initialize the enabled value for the type of report being generated
    mbRunType(eRunType.Power) = _
            (frmTPCHMetrics.chkRun(eRunType.Power).Value = vbChecked)
    mbRunType(eRunType.Throughput) = _
            (frmTPCHMetrics.chkRun(eRunType.Throughput).Value = vbChecked)

    If (Not mbRunType(eRunType.Power) And Not
mbRunType(eRunType.Throughput)) Then
        MsgBox "Atleast one report type must be selected."
        Exit Sub
    End If

    ' Initialize the stream identifier bounds depending on the type of
    ' report being generated
    miStreamIdLo = IIf(mbRunType(eRunType.Power), miStreamPowerRun,
miStreamPowerRun + 1)
    miStreamIdHi = IIf(mbRunType(eRunType.Throughput), miStreamIdMax,
miStreamPowerRun)
    If mbRunType(eRunType.Throughput) Then
        ' Initialize the number of throughput streams if a throughput
        ' metric is being generated
        sTemp = Trim(frmTPCHMetrics.cboStreams.Text)
        If sTemp = gsEmptyString Then
            MsgBox "Please enter the number of throughput streams that were
executed."
            Exit Sub
        End If

        If IsNumeric(sTemp) = False Then
            MsgBox "The number of throughput streams must be numeric."
            Exit Sub
        End If

        miStreamIdHi = CInt(sTemp)
        If miStreamIdHi < 1 Or miStreamIdHi > miStreamIdMax Then
            MsgBox "Number of throughput streams must range from 1 to 10. " & _
                    "Disable the throughput run checkbox if no throughput streams were
executed."
            Exit Sub
        End If
    End If

    ' Initialize the run identifier
    sTemp = Trim(frmTPCHMetrics.cboRunId.Text)
```

```
    If sTemp = gsEmptyString Then
        MsgBox "Please enter the run identifier."
        Exit Sub
    End If

    If IsNumeric(sTemp) = False Then
        MsgBox "The run identifier must be numeric."
        Exit Sub
    End If
    mlRunId = CLng(sTemp)

    ' Initialize the scale factor for the run
    sTemp = Trim(frmTPCHMetrics.cboSF.Text)
    If sTemp = gsEmptyString Then
        MsgBox "Please enter the scale factor for the run."
        Exit Sub
    End If

    If IsNumeric(sTemp) = False Then
        MsgBox "The scale factor must be numeric."
        Exit Sub
    End If
    mfSF = CSng(sTemp)

    ' Initialize the workspace definition file
    msDbFile = Trim(frmTPCHMetrics.txtDbFile)
    If msDbFile = gsEmptyString Then
        MsgBox "Please enter the workspace definition file (.stp) containing the run
history."
        Exit Sub
    End If

    ' Initialize the report file and ensure that it doesn't already exist
    msOutput = Trim(frmTPCHMetrics.txtReport)
    If msOutput = gsEmptyString Then
        MsgBox "Please enter the output file name."
        Exit Sub
    End If

    msOutput = MakePathValid(msOutput)
    If msOutput = gsEmptyString Then
        MsgBox "The output file path is not valid. Please enter a valid path."
        Exit Sub
    End If

    If Dir$(msOutput, vbNormal + vbHidden + vbReadOnly + vbSystem) <>
gsEmptyString Then
        MsgBox "The output file already exists. Please delete it or enter a new one."
        Exit Sub
    End If

    ' Initialize the number of insert and delete segments per update set
    sTemp = Trim(frmTPCHMetrics.txtInsertSegments.Text)
    If sTemp = gsEmptyString Then
        MsgBox "Please enter the number of insert segments per update set.
(StepMaster parameter: INSERT_SEGMENTS_PER_UPDATE_SET)"
        Exit Sub
    End If

    If IsNumeric(sTemp) = False Then
        MsgBox "The number of insert segments must be numeric. (StepMaster
parameter: INSERT_SEGMENTS_PER_UPDATE_SET)"
        Exit Sub
    End If
    miInsSegmentsPerUpdateSet = CInt(sTemp)

    sTemp = Trim(frmTPCHMetrics.txtDeleteSegments.Text)
    If sTemp = gsEmptyString Then
        MsgBox "Please enter the number of delete segments per update set.
(StepMaster parameter: DELETE_SEGMENTS_PER_UPDATE_SET)"
        Exit Sub
    End If

    If IsNumeric(sTemp) = False Then
        MsgBox "The number of delete segments must be numeric. (StepMaster
parameter: DELETE_SEGMENTS_PER_UPDATE_SET)"
        Exit Sub
    End If
    miDelSegmentsPerUpdateSet = CInt(sTemp)

#If Not TPCH_KIT Then
    If mbRunType(eRunType.Throughput) Then
        ' Initialize the value of the first update set
        sTemp = Trim(frmTPCHMetrics.txtUpdateSetFirst.Text)
        If sTemp = gsEmptyString Then
            MsgBox "Please enter the first update set for the throughput run.
(StepMaster parameter: 1ST_THRUPUT_UPDATE_SET)"
            Exit Sub
        End If

        If IsNumeric(sTemp) = False Then
            MsgBox "The first update set for the throughput run must be numeric.
(StepMaster parameter: 1ST_THRUPUT_UPDATE_SET)"
            Exit Sub
        End If
        miUpdateSetFirst = CInt(sTemp)
    End If
#End If

    ' The user inputs are valid. Call a procedure to query step execution
    ' details and generate the TPCHMetric report.
    Call QueryRunDetails

    Exit Sub

GenerateSelectedErr:
    DisplayErrors

End Sub
Private Function LabelStartPower(iQueryId As Integer) As String
    ' Returns the step label for the passed in power query identifier
    ' Uses the label that marks the start of execution for the query
    ' E.g. TPC-H Kit, Stream1, RF1: the start time for the refresh function
    ' is the start time for the step, 'Throughput - Semaphore Loop for RF Delay'
    ' but the end time for the refresh function is the end time for the step,
    ' 'Throughput - Execute Stream1 RF1'
    ' Similarly, in the HP version of the TPC-H Kit, Throughput Run, RF1:
    ' the start time for RF1 is the start time for the step,
    ' 'Wait for Throughput Run' but the end time for RF1 is the end time
    ' for the step, 'Throughput RF1'
    ' E.g. in the HP version of the TPC-H Kit, query 15 is executed in
    ' 2 parts. Therefore, the start time for query 15 is the start time
    ' for the step, 'Create REVENUE View' but the end time for query 15
    ' is the end time for the step, 'Execute using REVENUE'

    Select Case iQueryId
        Case miQueryIdRF1
            LabelStartPower = msStartStepLabelPowerRF1

        Case miQueryIdRF2
            LabelStartPower = msStartStepLabelPowerRF2

    #If Not TPCH_KIT Then
        Case miQuery15
            LabelStartPower = msStartStepLabelPower15
    #End If

        Case Else
            LabelStartPower = msStartStepLabelPowerOther & Format$(iQueryId,
msFmtNumberPadWZeroes)

    End Select

End Function
Private Function LabelEndPower(iQueryId As Integer) As String
    ' Returns the step label for the passed in power query identifier
    ' Uses the label that marks the end of execution for the query
    ' E.g. TPC-H Kit, Stream1, RF1: the start time for the refresh function
```

```
    ' is the start time for the step, 'Throughput - Semaphore Loop for RF Delay'
    ' but the end time for the refresh function is the end time for the step,
    ' 'Throughput - Execute Stream1 RF1'
    ' Similarly, in the HP version of the TPC-H Kit, Throughput Run, RF1:
    ' the start time for RF1 is the start time for the step,
    ' 'Wait for Throughput Run' but the end time for RF1 is the end time
    ' for the step, 'Throughput RF1'
    ' E.g. in the HP version of the TPC-H Kit, query 15 is executed in
    ' 2 parts. Therefore, the start time for query 15 is the start time
    ' for the step, 'Create REVENUE View' but the end time for query 15
    ' is the end time for the step, 'Execute using REVENUE'

    #If TPCH_KIT Then
        Debug.Assert iQueryId = miQueryIdRF1 Or iQueryId = miQueryIdRF2
        Select Case iQueryId
            Case miQueryIdRF1
                LabelEndPower = msEndStepLabelPowerRF1
            Case miQueryIdRF2
                LabelEndPower = msEndStepLabelPowerRF2
        End Select
    #Else
        Debug.Assert iQueryId = miQueryIdRF1 Or iQueryId = miQueryIdRF2 Or
iQueryId = miQuery15
        Select Case iQueryId
            Case miQueryIdRF1
                LabelEndPower = msEndStepLabelPowerRF1
            Case miQueryIdRF2
                LabelEndPower = msEndStepLabelPowerRF2
            Case miQuery15
                LabelEndPower = msEndStepLabelPower15
        End Select
    #End If

End Function
#If Not TPCH_KIT Then
Private Function GetEndStepItThroughput(iQueryId As Integer) As String
    ' Returns the iterator value for the passed in query identifier
    ' for the throughput run
    ' Returns the iterator value that marks the end of execution for the query
    ' E.g. in the HP version of the TPC-H Kit, query 15 is executed in
    ' 2 parts. Therefore, the iterator value for the start of query 15
    ' is 15V but the the iterator value for the end of query 15 is 15

    Debug.Assert iQueryId = miQuery15
    GetEndStepItThroughput = miQuery15

End Function
#End If

Private Function LabelStartThroughput(iStream As Integer, _
        iQuery As Integer) As String
    ' Returns the step label for the passed in query and stream in the
    ' throughput run
    ' Returns the label that marks the start of execution for the query
    ' E.g. TPC-H Kit, Stream1, RF1: the start time for the refresh function
    ' is the start time for the step, 'Throughput - Semaphore Loop for RF Delay'
    ' but the end time for the refresh function is the end time for the step,
    ' 'Throughput - Execute Stream1 RF1'
    ' Similarly, in the HP version of the TPC-H Kit, Throughput Run, RF1:
    ' the start time for RF1 is the start time for the step,
    ' 'Wait for Throughput Run' but the end time for RF1 is the end time
    ' for the step, 'Throughput RF1'

    #If TPCH_KIT Then
        Select Case iQuery
            Case miQueryIdRF1
                LabelStartThroughput = IIf(iStream = miStreamPowerRun + 1, _
                        msStartStepLabelThroughputRF1First, _
                        msLblThroughputRF1)

            Case miQueryIdRF2
                LabelStartThroughput = msLblThroughputRF2

            Case Else
                LabelStartThroughput = msStepLabelThroughputOther & _
                        Format$(iStream, msFmtNumberWOZeroes)
        End Select
    #Else
        Select Case iQuery
            Case miQueryIdRF1
                LabelStartThroughput = msStartStepLabelThroughputRF1

            Case miQueryIdRF2
                LabelStartThroughput = msStartStepLabelThroughputRF2

            Case Else
                LabelStartThroughput = msStartStepLabelThroughputOther & _
                        Format$(iStream, msFmtNumberPadWZeroes)
        End Select
    #End If

End Function
Private Function LabelEndThroughput(iStream As Integer, iQuery As Integer) As
String
    ' Returns the step label for the passed in query and stream in the
    ' throughput run
    ' Returns the label that marks the end of execution for the query
    ' E.g. TPC-H Kit, Stream1, RF1: the start time for the refresh function
    ' is the start time for the step, 'Throughput - Semaphore Loop for RF Delay'
    ' but the end time for the refresh function is the end time for the step,
    ' 'Throughput - Execute Stream1 RF1'
    ' Similarly, in the HP version of the TPC-H Kit, Throughput Run, RF1:
    ' the start time for RF1 is the start time for the step,
    ' 'Wait for Throughput Run' but the end time for RF1 is the end time
    ' for the step, 'Throughput RF1'

    #If TPCH_KIT Then
        Select Case iQuery
            Case miQueryIdRF1
                LabelEndThroughput = msLblThroughputRF1

            Case miQueryIdRF2
                LabelEndThroughput = IIf(iStream = miStreamIdHi, _
                        msLblThroughputRF2, msEndStepLabelThroughputRF2Rest)

            Case Else
                Debug.Assert True
        End Select

    #Else
        LabelEndThroughput = msEndStepLabelThroughputRF1
    #End If

End Function

Private Function IteratorThroughput(iStream As Integer, iQuery As Integer) As
String
    ' Returns the iterator value for the passed in stream and query
    ' for the throughput run
    ' Returns the iterator value that marks the start of execution for the query
    ' E.g. in the HP version of the TPC-H Kit, query 15 is executed in
    ' 2 parts. Therefore, the iterator value for the start of query 15
    ' is 15V but the the iterator value for the end of query 15 is 15

    #If TPCH_KIT Then
        Dim iQIndex As Integer
    #End If

        Select Case iQuery
            Case miQueryIdRF1
                IteratorThroughput = msFirstRFIterator

            Case miQueryIdRF2
                IteratorThroughput = msFirstRFIterator

            Case Else
                #If TPCH_KIT Then
                    ' The step iterates from 1 to 22 but the queries are generated
```

```
            ' by qgen in the order that they should be executed in.
            ' e.g. Stream1Q1.sql contains the query corresponding to
            ' Stream 1, Query 21
            For iQIndex = miQueryIdMin To miQueryIdMax Step 1
               If arQuerySequence(iStream, iQIndex) = iQuery Then
                  Exit For
               End If
            Next iQIndex

            IteratorThroughput = Format$(iQIndex, msFmtNumberWOZeroes)
         #Else
            IteratorThroughput = Format$(iQuery, msFmtNumberWOZeroes)
            If iQuery = miQuery15 Then
               ' Query 15 starts with the creation of the view
               ' The query file is 15V + StreamId.sql and the iterator
               ' value is 15V
               IteratorThroughput = IteratorThroughput & "V"
            End If
         #End If
      End Select

End Function
Private Function IteratorPowerRF(iStream As Integer, iQuery As Integer) As
String
   ' Returns the iterator value for the passed in query in the PowerRun

   Debug.Assert iQuery = miQueryIdRF1 Or iQuery = miQueryIdRF2

   Select Case iQuery
      Case miQueryIdRF1
         IteratorPowerRF = msFirstRFIterator

      Case miQueryIdRF2
         IteratorPowerRF = msFirstRFIterator

   End Select

End Function

Private Sub CollectStats()
   ' This function determines the start time, end time and start time for
   ' the next step (if applicable) for each query in each stream

   Dim iStreamId As Integer
   Dim lParentInstId As Long
   Dim iQueryId As Integer
   Dim iNextQuery As Integer
   Dim iNextStream As Integer
   Dim sLabel As String
   Dim sIt As String
   Dim sSql As String
   Dim iUBound As Integer

   If mbRunType(eRunType.Power) Then
      ' Populate start and end times for the PowerRun
      iStreamId = miStreamPowerRun

      For iQueryId = miQueryIdMin To miQueryIdMax Step 1
         ' Determine the start time for the step using the step label
         sLabel = LabelStartPower(iQueryId)

         sSql = "step_label = " & MakeStringFieldValid(sLabel)

         If (iQueryId = miQueryIdRF1 Or iQueryId = miQueryIdRF2) Then
            sIt = IteratorPowerRF(iStreamId, iQueryId)
            sSql = sSql & " and iterator_value = " & MakeStringFieldValid(sIt)
         End If

         mrecDtls.FindFirst sSql
         If mrecDtls.NoMatch = True Then
            MsgBox "Step " & "'" & sLabel & "'" & " not found. " & _
                  "The PowerRun was not completely executed!"
            Exit Sub
         End If
         arExecutionDtls(iStreamId, iQueryId).StartTime =
mrecDtls.Fields("start_time").Value

         ' Determine the end time for the step

         If iQueryId = miQueryIdRF1 Or iQueryId = miQueryIdRF2 Then
            ' RF1 and RF2, find the time at which the last refresh function
            ' completed
            sLabel = LabelEndPower(iQueryId)
            arExecutionDtls(iStreamId, iQueryId).EndTime = 0

            iUBound = IIf(iQueryId = miQueryIdRF1,
miInsSegmentsPerUpdateSet, _
                  miDelSegmentsPerUpdateSet)

            For iNextQuery = msFirstRFIterator To iUBound Step 1
               sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
                     " and iterator_value = "
               sIt = Format$(iNextQuery, msFmtNumberWOZeroes)
               sSql = sSql & MakeStringFieldValid(sIt)
               mrecDtls.FindFirst sSql
               If mrecDtls.NoMatch = True Then
                  MsgBox "Step " & "'" & sLabel & "'" & _
                        " not found for iterator " & "'" & sIt & "'" & ". " & _
                        "The PowerRun was not completely executed!"
                  Exit Sub
               End If
               If mrecDtls.Fields("end_time").Value > _
                     arExecutionDtls(iStreamId, iQueryId).EndTime Then
                  arExecutionDtls(iStreamId, iQueryId).EndTime = _
                     mrecDtls.Fields("end_time").Value
               End If
            Next iNextQuery
         #If Not TPCH_KIT Then
         ElseIf iQueryId = miQuery15 Then
            sLabel = LabelEndPower(iQueryId)
            sSql = "step_label = " & MakeStringFieldValid(sLabel)
            mrecDtls.FindFirst sSql
            If mrecDtls.NoMatch = True Then
               MsgBox "Step " & "'" & sLabel & "'" & " not found. " & _
                     "The PowerRun was not completely executed!"
               Exit Sub
            End If
            arExecutionDtls(iStreamId, iQueryId).EndTime =
mrecDtls.Fields("end_time").Value
         #End If
         Else
            arExecutionDtls(iStreamId, iQueryId).EndTime =
mrecDtls.Fields("end_time").Value
         End If

         ' The timing interval for a query is the time between the start
         ' time of the next query and the start time of the query, except
         ' for the last query of the set for which it is the time between
         ' the end time of the query and the start time
         ' The timing interval for the execution of the refresh function
         ' must be measured between the start time for the first
         ' refresh function and the end time for the last refresh function
         ' Refer to Sections 5.3.7.2 and 5.3.7.3 of the TPC-H Spec. Rev 1.1.0
         If NextStepExists(iStreamId, iQueryId, iNextQuery) Then
            ' Determine the start time of the next step, if applicable
            sLabel = LabelStartPower(iNextQuery)

            sSql = "step_label = " & MakeStringFieldValid(sLabel)
            mrecDtls.FindFirst sSql
            If mrecDtls.NoMatch = True Then
               MsgBox "Step " & "'" & sLabel & "'" & " not found. " & _
                     "The PowerRun was not completely executed!"
               Exit Sub
            End If

            arExecutionDtls(iStreamId, iQueryId).StartTimeNextStep =
mrecDtls.Fields("start_time").Value
```

```
        Else
            ' Else, initialize start time for the next step to the end time for the step
            arExecutionDtls(iStreamId, iQueryId).StartTimeNextStep = _
                arExecutionDtls(iStreamId, iQueryId).EndTime
        End If

        ' Convert all the 64-bit timestamps to string values
        arExecutionDtls(iStreamId, iQueryId).sStartTm =
JulianToString(arExecutionDtls(iStreamId, iQueryId).StartTime)
        arExecutionDtls(iStreamId, iQueryId).sEndTm =
JulianToString(arExecutionDtls(iStreamId, iQueryId).EndTime)
        arExecutionDtls(iStreamId, iQueryId).sStartNextTm =
JulianToString(arExecutionDtls(iStreamId, iQueryId).StartTimeNextStep)
    Next iQueryId
  End If

  If mbRunType(eRunType.Throughput) Then
    ' Populate start and end times for the ThroughputRun
    For iStreamId = miStreamIdMin + 1 To miStreamIdHi

        For iQueryId = miQueryIdMin To miQueryIdMax Step 1
            ' Determine the start time for the step using the step label
            sLabel = LabelStartThroughput(iStreamId, iQueryId)
            sSql = "step_label = " & MakeStringFieldValid(sLabel)

            #If TPCH_KIT Then
                If (iQueryId <> miQueryIdRF1 Or iStreamId <> miStreamPowerRun
+ 1) Then
                    sIt = IteratorThroughput(iStreamId, iQueryId)
                    sSql = sSql & " and iterator_value = " &
MakeStringFieldValid(sIt)
                End If
            #Else
                If iQueryId <> miQueryIdRF1 Then
                    sIt = IteratorThroughput(iStreamId, iQueryId)
                    sSql = sSql & " and iterator_value = " &
MakeStringFieldValid(sIt)
                End If
            #End If

            If iQueryId = miQueryIdRF1 Or iQueryId = miQueryIdRF2 Then
                lParentInstId = ParentInstId(iStreamId, iQueryId)
                If lParentInstId = 0 Then Exit Sub
                sSql = sSql & " and parent_instance_id = " & CStr(lParentInstId)
            End If

            mrecDtls.FindFirst sSql
            If mrecDtls.NoMatch = True Then
                MsgBox "Step " & "'" & sLabel & "'" & _
                    " not found for iterator " & "'" & sIt & "'" & ". " & _
                    "The Throughput Run was not completely executed!"
                Exit Sub
            End If

            arExecutionDtls(iStreamId, iQueryId).StartTime =
mrecDtls.Fields("start_time").Value

            ' Determine the end time for the step

            If iQueryId = miQueryIdRF1 Or iQueryId = miQueryIdRF2 Then
                ' RF1 and RF2, find the time at which the last refresh
                ' function completed
                arExecutionDtls(iStreamId, iQueryId).EndTime = 0

                iUBound = IIf(iQueryId = miQueryIdRF1,
miInsSegmentsPerUpdateSet, _
                    miDelSegmentsPerUpdateSet)

                #If TPCH_KIT Then
                    sLabel = LabelEndThroughput(iStreamId, iQueryId)
                #Else
                    If iQueryId = miQueryIdRF1 Then
                        sLabel = LabelEndThroughput(iStreamId, iQueryId)
                    End If
                #End If
                lParentInstId = ParentInstIdEnd(iStreamId, iQueryId)

                #If TPCH_KIT Then
                    If iQueryId = miQueryIdRF2 And iStreamId <> miStreamIdHi
Then
                        sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
                            " and parent_instance_id = " & CStr(lParentInstId)

                        mrecDtls.FindFirst sSql
                        If mrecDtls.NoMatch = True Then
                            MsgBox "Step " & "'" & sLabel & "'" & _
                                " not found for iterator " & "'" & sIt & "'" & ". " & _
                                "The Throughput run was not completely executed!"
                            Exit Sub
                        End If
                        arExecutionDtls(iStreamId, iQueryId).EndTime = _
                            mrecDtls.Fields("end_time").Value
                    Else
                #End If

                    ' Find the time at which the last refresh function completed
                    For iNextQuery = msFirstRFIterator To iUBound Step 1
                        sIt = Format$(iNextQuery, msFmtNumberWOZeroes)
                        sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
                            " and iterator_value = " & MakeStringFieldValid(sIt)
                        sSql = sSql & " and parent_instance_id = " &
CStr(lParentInstId)

                        mrecDtls.FindFirst sSql
                        If mrecDtls.NoMatch = True Then
                            MsgBox "Step " & "'" & sLabel & "'" & _
                                " not found for iterator " & "'" & sIt & "'" & ". " & _
                                "The Throughput run was not completely executed!"
                            Exit Sub
                        End If
                        If mrecDtls.Fields("end_time").Value > _
                            arExecutionDtls(iStreamId, iQueryId).EndTime Then
                            arExecutionDtls(iStreamId, iQueryId).EndTime = _
                                mrecDtls.Fields("end_time").Value
                        End If
                    Next iNextQuery

                #If TPCH_KIT Then
                    End If
                #End If

                #If Not TPCH_KIT Then
                ElseIf iQueryId = miQuery15 Then
                    sLabel = LabelStartThroughput(iStreamId, iQueryId)
                    sIt = GetEndStepItThroughput(iQueryId)
                    sSql = "step_label = " & MakeStringFieldValid(sLabel) & _
                        " and iterator_value = " & MakeStringFieldValid(sIt)

                    mrecDtls.FindFirst sSql
                    If mrecDtls.NoMatch = True Then
                        MsgBox "Step " & "'" & sLabel & "'" & _
                            " not found for iterator " & "'" & sIt & "'" & ". " & _
                            "The Throughput run was not completely executed!"
                        Exit Sub
                    End If
                    arExecutionDtls(iStreamId, iQueryId).EndTime =
mrecDtls.Fields("end_time").Value
                #End If
            Else
                arExecutionDtls(iStreamId, iQueryId).EndTime =
mrecDtls.Fields("end_time").Value
            End If

            ' The timing interval for a query is the time between the start
            ' time of the next query and the start time of the query, except
            ' for the last query of the set for which it is the time between
            ' the end time of the query and the start time
            ' The timing interval for the execution of the refresh function
```

```
            ' must be measured between the start time for the first
            ' refresh function and the end time for the last refresh function
            ' Refer to Sections 5.3.7.2 and 5.3.7.3 of the TPC-H Spec. Rev 1.1.0
            If NextStepExists(iStreamId, iQueryId, iNextQuery) Then
               ' Determine the start time of the next step, if applicable
               iNextStream = iStreamId
               sLabel = LabelStartThroughput(iNextStream, iNextQuery)
               sSql = "step_label = " & MakeStringFieldValid(sLabel)
               #If TPCH_KIT Then
                  sIt = IteratorThroughput(iNextStream, iNextQuery)
                  sSql = sSql & " and iterator_value = " &
MakeStringFieldValid(sIt)
               #Else
                  sIt = IteratorThroughput(iNextStream, iNextQuery)
                  sSql = sSql & " and iterator_value = " &
MakeStringFieldValid(sIt)
               #End If

               If iQueryId = miQueryIdRF1 Or iQueryId = miQueryIdRF2 Then
                  lParentInstId = ParentInstId(iNextStream, iNextQuery)
                  If lParentInstId = 0 Then Exit Sub
                  sSql = sSql & " and parent_instance_id = " & CStr(lParentInstId)
               End If

               mrecDtls.FindFirst sSql
               If mrecDtls.NoMatch = True Then
                  MsgBox "Step " & "'" & sLabel & "'" & _
                        " not found for iterator " & "'" & sIt & "'" & ". " & _
                        "The Throughput run was not completely executed!"
                  Exit Sub
               End If

               arExecutionDtls(iStreamId, iQueryId).StartTimeNextStep =
mrecDtls.Fields("start_time").Value
            Else
               ' Else, initialize start time for the next step to the end time for the step
               arExecutionDtls(iStreamId, iQueryId).StartTimeNextStep =
arExecutionDtls(iStreamId, iQueryId).EndTime
            End If

            ' Convert all the 64-bit timestamps to string values
            arExecutionDtls(iStreamId, iQueryId).sStartTm =
JulianToString(arExecutionDtls(iStreamId, iQueryId).StartTime)
            arExecutionDtls(iStreamId, iQueryId).sEndTm =
JulianToString(arExecutionDtls(iStreamId, iQueryId).EndTime)
            arExecutionDtls(iStreamId, iQueryId).sStartNextTm =
JulianToString(arExecutionDtls(iStreamId, iQueryId).StartTimeNextStep)

         Next iQueryId
      Next iStreamId
   End If

   ' Call a procedure to calculate the elapsed time for each step and
   ' the Power and Throughput Metrics
   Call CalcElapsedTmAndPrint

End Sub
Private Function JulianToString(cJTime As Currency)
   ' Converts the passed in 64-bit date to a formatted string that can
   ' be printed in the report
   ' The format of date/time in reports is yyyy-mm-dd hh:mm:ss.sss
   ' This format has these benefits:
   '  - is sortable, since all fields are arranged most to least significant
   '    and all numeric
   '  - is directly interpretted by Excel as a time/date value
   '  - is fixed width
   '  - is Y2K compliant

   Dim lYr As Long
   Dim lMon As Long
   Dim lDt As Long
   Dim lHr As Long
   Dim lMin As Long
   Dim lSec As Long
   Dim lMs As Long
   Dim cJTemp As Currency

   Call smtimelib.JulianToTime(cJTime, lYr, lMon, lDt, lHr, lMin, lSec, lMs)
   JulianToString = Format$(lYr, msFmtYrPadWZeroes) & msDtSeparator & _
         Format$(lMon, msFmtNumberPadWZeroes) & msDtSeparator & _
         Format$(lDt, msFmtNumberPadWZeroes) & msBlank & _
         Format$(lHr, msFmtNumberPadWZeroes) & msTmSeparator & _
         Format$(lMin, msFmtNumberPadWZeroes) & msTmSeparator & _
         Format$(lSec, msFmtNumberPadWZeroes) & msMsSeparator & _
         Format$(lMs, msFmtMsPadWZeroes)

End Function
Public Sub LoadRunIds(cboLoad As ComboBox, sDbFile As String)
   ' Opens the passed in workspace definition file and loads all the
   ' run identifiers in the passed in combo box

   Dim strSql As String
   Dim lRunId As Long

   On Error GoTo LoadRunIdsErr

   Screen.MousePointer = vbHourglass

   Set mwrkJet = CreateWorkspace("tpch", "admin", gsEmptyString, dbUseJet)

   ' Open database for exclusive access
   Set mDB = mwrkJet.OpenDatabase(sDbFile, Options:=True)

   strSql = "select run_id from run_header order by run_id desc"
   Set mrecDtls = mDB.OpenRecordset(strSql, dbOpenSnapshot)

   cboLoad.Clear

   If mrecDtls.RecordCount = 0 Then
      Screen.MousePointer = vbDefault
      MsgBox "There are no run records. Unable to load list of run ids."
   Else
      mrecDtls.MoveFirst
      While Not mrecDtls.EOF
         ' Add run identifier to the combo box
         lRunId = mrecDtls.Fields("run_id").Value
         cboLoad.AddItem CStr(lRunId)

         mrecDtls.MoveNext
      Wend
      If cboLoad.ListCount > 0 Then
         cboLoad.ListIndex = 0
      End If
   End If

   mrecDtls.Close
   CloseDbAndWsp

   Screen.MousePointer = vbDefault

   Exit Sub

LoadRunIdsErr:

   Screen.MousePointer = vbDefault
   DisplayErrors

   CloseDbAndWsp

End Sub
Public Sub LoadSFAndStreams(cboSF As ComboBox, cboStreams As
ComboBox)
   ' Loads all the valid scale factors in cboSF
   ' Loads the number of throughput streams for each scale factor in
   ' cboStreams
   ' This procedure assumes that the arrays, marrSF and marrStreams have
   ' already been initialized.
```

```
    Dim iIndex As Integer

    On Error GoTo LoadSFAndStreamsErr

    cboSF.Clear
    cboStreams.Clear

    For iIndex = LBound(marrSF) To UBound(marrSF)
        cboSF.AddItem CStr(marrSF(iIndex))
        cboSF.ItemData(iIndex) = iIndex

        cboStreams.AddItem CStr(marrStreams(iIndex))
        cboStreams.ItemData(iIndex) = iIndex
    Next iIndex

    cboSF.ListIndex = miDefaultSF
    cboStreams.ListIndex = miDefaultSF

    Exit Sub

LoadSFAndStreamsErr:

    DisplayErrors

End Sub

Private Function NextStepExists(iStream As Integer, iQuery As Integer, _
        iNextQuery As Integer) As Boolean
    ' Returns False if this is the last step in the query stream
    ' If not, populates iNextQuery with the index of the next query

    Dim iQIndex As Integer

    If iQuery = miQueryIdRF1 Or iQuery = miQueryIdRF2 Then
        NextStepExists = False
    Else
        For iQIndex = LBound(arQuerySequence, miDimensionQuery) _
                To UBound(arQuerySequence, miDimensionQuery) Step 1
            If arQuerySequence(iStream, iQIndex) = iQuery Then
                Exit For
            End If
        Next iQIndex

        NextStepExists = (iQIndex < UBound(arQuerySequence,
miDimensionQuery))

        If NextStepExists Then
            iNextQuery = arQuerySequence(iStream, iQIndex + 1)
        End If
    End If

End Function
Private Sub PrintStatistics(fQppPower As Double, fElapsedThroughput As
Double, _
        fQppThroughput As Double, fQppH As Double, sStartTm As String, _
        sEndTm As String)
    ' This function prints the execution statistics for each step
    ' and the Power and Throughput Metrics to the output file

    Dim iQIndex As Integer
    Dim iSIndex As Integer
    Dim sPrint As String
    Dim hOutputFile As Integer

    hOutputFile = 0

    hOutputFile = FreeFile
    Open msOutput For Output Shared As hOutputFile
    ' Header information
    Print #hOutputFile, "Workspace file:" & vbTab & msDbFile
    Print #hOutputFile, "Run identifier:" & vbTab & CStr(mlRunId)
    Print #hOutputFile, "Database Scale Factor:" & vbTab & CStr(mfSF)
    Print #hOutputFile, "Query Streams for Throughput Test:" & vbTab &
CStr(miStreamIdHi)
    Print #hOutputFile, "Measurement Interval in Throughput Test (Ts):" & vbTab
& CStr(fElapsedThroughput) & vbTab & vbTab & _
                "Start Time: " & vbTab & sStartTm & vbTab & vbTab & _
                "End Time: " & vbTab & sEndTm
    Print #hOutputFile, "TPC-H Power:" & vbTab & CStr(fQppPower)
    Print #hOutputFile, "TPC-H Throughput:" & vbTab & CStr(fQppThroughput)
    Print #hOutputFile, "TPC-H Composite Query-Per-Hour Rating (QphH):" &
vbTab & CStr(fQppH)
    Print #hOutputFile,

    ' Print Query names in column header
    Print #hOutputFile, vbTab;

    For iQIndex = miQueryIdMin To miQueryIdRF1 - 1
        sPrint = "Q" & Format$(iQIndex, msFmtNumberWOZeroes) & vbTab
        Print #hOutputFile, sPrint;
    Next iQIndex

    For iQIndex = miQueryIdRF1 To miQueryIdRF2
        sPrint = "RF" & Format$(iQIndex - miQueryIdRF1 + 1,
msFmtNumberWOZeroes) & vbTab
        Print #hOutputFile, sPrint;
    Next iQIndex

    Print #hOutputFile,

    ' Print TPCH timing intervals for each query in timing grid
    For iSIndex = miStreamIdLo To miStreamIdHi

        sPrint = "Stream " & Format$(iSIndex, msFmtNumberPadWZeroes) &
vbTab
        Print #hOutputFile, sPrint;

        For iQIndex = miQueryIdMin To miQueryIdMax Step 1

            sPrint = CStr(arExecutionDtls(iSIndex, iQIndex).ElapsedTpch) & vbTab
            Print #hOutputFile, sPrint;
        Next iQIndex

        Print #hOutputFile,

    Next iSIndex

    Print #hOutputFile,
    Print #hOutputFile, "Elapsed times for each step"
    Print #hOutputFile,

    ' Print elapsed times for each query in timing grid
    For iSIndex = miStreamIdLo To miStreamIdHi

        sPrint = "Stream " & Format$(iSIndex, msFmtNumberPadWZeroes) &
vbTab
        Print #hOutputFile, sPrint;

        For iQIndex = miQueryIdMin To miQueryIdMax Step 1

            sPrint = CStr(arExecutionDtls(iSIndex, iQIndex).ElapsedTime) & vbTab
            Print #hOutputFile, sPrint;
        Next iQIndex

        Print #hOutputFile,

    Next iSIndex

    Print #hOutputFile,
    Print #hOutputFile, vbTab & "Query Id" & vbTab & "Start Time" & vbTab & _
            "End Time" & vbTab & "Start Next Step" & vbTab & "Step Delta" &
vbTab & _
            "Official Query Elapsed" & vbTab & "Difference"

    ' Print step execution details, viz. start time, end time, TPCH timing
    ' interval and elapsed times for each query
    For iSIndex = miStreamIdLo To miStreamIdHi
```

```
      sPrint = "Stream " & Format$(iSIndex, msFmtNumberPadWZeroes)
      Print #hOutputFile, sPrint;

      For iQIndex = miQueryIdMin To miQueryIdMax Step 1

         sPrint = vbTab & CStr(iQIndex) & vbTab & _
               arExecutionDtls(iSIndex, iQIndex).sStartTm & vbTab & _
               arExecutionDtls(iSIndex, iQIndex).sEndTm & vbTab & _
               arExecutionDtls(iSIndex, iQIndex).sStartNextTm & vbTab & _
               CStr(arExecutionDtls(iSIndex, iQIndex).ElapsedTime) & vbTab & _
               CStr(arExecutionDtls(iSIndex, iQIndex).ElapsedTpch) & vbTab & _
               Round(CStr(arExecutionDtls(iSIndex, iQIndex).ElapsedTpch -
arExecutionDtls(iSIndex, iQIndex).ElapsedTime), miNumDecimalPlaces)
         Print #hOutputFile, sPrint
      Next iQIndex

      Print #hOutputFile,

   Next iSIndex

   Close hOutputFile
   hOutputFile = 0

   MsgBox "The report has been generated at " & gsSQ & msOutput & gsSQ &
"!"

   Exit Sub

GenerateSelectedErr:
   DisplayErrors

   If hOutputFile <> 0 Then
      Close hOutputFile
   End If

End Sub
Public Sub SFChanged()

   ' Called when the scale factor is changed - modifies the number
   ' of streams depending on the scale factor
   frmTPCHMetrics.cboStreams.ListIndex = frmTPCHMetrics.cboSF.ListIndex

End Sub
```

Utility.bas

```
Attribute VB_Name = "Utility"
'  FILE:      Utility.bas
'             Reporting Program for Microsoft TPC-H Kit Ver. 1.00
'             Copyright Microsoft, 1999
'             All Rights Reserved
'
'
'  PURPOSE:    Utility functions for TPCHMetrics report generation program
'  Contact:    Reshma Tharamal (reshmat@microsoft.com)
'
Option Explicit

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
   lpstrFileTitle As String
   nMaxFileTitle 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

Public Const MAX_PATH = 255
Public Const gsEmptyString As String = ""
Public Const gsSQ As String = """
Public Const gsPeriod As String = "."
Public Const gsDefDBFileExt As String = ".stp"
Public Const gsFileSeparator = "\"
Public Const gsPipe = "|"
Public Const gsNullValue = "null"

Private Const ms_DLG_TITLE_OPEN = "Open"

Private Enum EOpenFile
   OFN_OVERWRITEPROMPT = &H2
   OFN_HIDEREADONLY = &H4
   OFN_FILEMUSTEXIST = &H1000
   OFN_EXPLORER = &H80000
End Enum

Private Const ml_FILE_DB_FLAGS = OFN_FILEMUSTEXIST
' Note: The flag below is meaningless, since it doesn't work for open file dialogs
Private Const ml_FILE_OUTPUT_FLAGS = OFN_OVERWRITEPROMPT

Private Const ms_FILE_ALL_FILTER = "|All Files (*.*)|*.*"
Private Const ms_FILE_DB_FILTER = "Stepmaster Workspace Files (*" &
gsDefDBFileExt & ")|*" & gsDefDBFileExt
Private Const ms_FILE_OUTPUT_FILTER = "Text Files (*.txt)|*.txt"
Public Function MakeStringFieldValid(strField As String) As String
   ' Returns a string that can be appended to any insert
   ' or modify (sql) statement

   ' It checks whether the text is empty
   ' If so, it returns the string, "null"
   If strField = gsEmptyString Then
      MakeStringFieldValid = gsNullValue
   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
      MakeStringFieldValid = ReplaceSubString(strField, gsSQ, gsSQ & gsSQ)

      ' Replace pipe characters with the corresponding chr function
      MakeStringFieldValid = ReplaceSubString(strField, gsPipe, "' & Chr(124) &
'")

      ' Enclose the string in single quotes
      MakeStringFieldValid = gsSQ & strField & gsSQ

   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

   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

End Function

Public Function ShowFileOpenDialog(ByVal strFilter As String, _
        ByVal strDialogTitle As String, ByVal lngFlags As Long, _
        Optional ByVal strOldFile As String = gsEmptyString) 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 ShowFileOpenDialogErr

    If Not strOldFile <> gsEmptyString Then
        intPos = InstrR(strOldFile, gsPeriod)
        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
        ShowFileOpenDialog = Left$(opfile.lpstrFile, InStr(opfile.lpstrFile,
vbNullChar) - 1)
    Else
        ShowFileOpenDialog = strOldFile
    End If

    Exit Function

ShowFileOpenDialogErr:
    Call DisplayErrors
    ' Reset the selection to the passed in file, if any
    ShowFileOpenDialog = strOldFile

End Function
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

    intPrev = intPos
    intPos = InStr(1, strMain, strSearch)

    Do While intPos > 0
        intPrev = intPos
        intPos = InStr(intPos + 1, strMain, strSearch)
    Loop
    InstrR = intPrev

End Function
Private Function MakeWindowsFilter(sFilter As String) As String

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

    ' 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

End Function
Public Function BrowseStpFile(ByVal strDefaultFile As String) 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

    On Error GoTo BrowseStpFileErr

    BrowseStpFile = ShowFileOpenDialog( _
        ms_FILE_DB_FILTER & ms_FILE_ALL_FILTER, _
        ms_DLG_TITLE_OPEN, _
        ml_FILE_DB_FLAGS, _
        strDefaultFile)

    Exit Function

BrowseStpFileErr:
    DisplayErrors
    BrowseStpFile = gsEmptyString

End Function
Public Function BrowseOutputFile(ByVal strDefaultFile As String) 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

    On Error GoTo BrowseOutputFileErr

    BrowseOutputFile = ShowFileOpenDialog( _
        ms_FILE_OUTPUT_FILTER & ms_FILE_ALL_FILTER, _
        ms_DLG_TITLE_OPEN, _
        ml_FILE_OUTPUT_FLAGS, _
        strDefaultFile)

    Exit Function

BrowseOutputFileErr:
    DisplayErrors
    BrowseOutputFile = gsEmptyString

End Function
Public Function MakePathValid(strFileName As String) As String
    ' Makes the passed in file path valid

    Dim strFileDir As String
    Dim strTempDir As String
    Dim strTempFile As String
    Dim sTemp As String
    Dim intPos As Integer
    Dim intStart As Integer

    strTempFile = strFileName
    intPos = InstrR(strFileName, gsFileSeparator)

    If intPos > 0 Then
        strFileDir = Left$(strTempFile, intPos - 1)
        If Dir$(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, gsFileSeparator)
        If Mid(strFileDir, 2, 1) = ":" Then
            ' Remove drive from directory specification
            intStart = InStr(intStart + 1, strFileDir, gsFileSeparator)
        End If

        Do While strTempDir <> strFileDir

            If intStart > 0 Then
                strTempDir = Left$(strFileDir, intStart - 1)
            Else
                strTempDir = strFileDir
            End If

            If 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, gsFileSeparator)
        Loop

        ' Sanity check
        If Dir$(strFileDir, vbDirectory) <> "" Then
            ' We were unable to create the file directory
            MsgBox "Unable to create directory " & strFileDir
            MakePathValid = gsEmptyString
        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

End Function
```

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 = &HFFFFFFFF
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
   lpstrFileTitle As String
   nMaxFileTitle 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" (lpsz 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 ShowFileOpenDialog(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 ShowFileOpenDialogErr

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

   Exit Function

ShowFileOpenDialogErr:
   Call LogErrors(Errors)
   ' Reset the selection to the passed in file, if any
   ShowFileOpenDialog = 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, tsql_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
```

Appendix F: Price Quotations

Microsoft Corporation
One Microsoft Way
Redmond, WA 98052-6399

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

Microsoft

February 7, 2008

IBM Corporation
Chris King
3079 Cornwallis Road
Durham, NC 27709

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
810-04779	**SQL Server 2005 Enterprise x64 Edition** *Server License with 25 CALs* *Discount Schedule: Open Program - Level C* *Unit Price reflects a 40% discount from the retail unit price of $13,969.*	$8,318	1	$8,318
359-01911	**SQL Server 2005 Client License** *Client Access License* *Discount Schedule: Open Program - No Level* *Unit Price reflects a 4% discount from the retail unit price of $163.*	$156	35	$5,460
P72-01684	**Windows Server 2003 R2 Enterprise x64 Edition** *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.*	$2,334	1	$2,334
127-00012	**Visual Studio Standard 2005** *Full License* *No Discount Applied*	$250	1	$250
N/A	**Microsoft Problem Resolution Services** *Professional Support* *(1 Incident)*	$245	1	$245

All products are currently orderable through Microsoft's normal distribution channels. A list of Microsoft's resellers can be found at **http://www.microsoft.com/products/info/render.aspx?view=22&type=mnp&content=22/licensing**

Defect support is included in the purchase price. Additional support is available from Microsoft PSS on an incident by incident basis at $245 per call.

This quote is valid for the next 90 days.

If we can be of any further assistance, please contact Jamie Reding at (425) 703-0510 or jamiere@microsoft.com.

Reference ID: PHchki0802070000005852.
Please include this Reference ID in any correspondence regarding this price quote.