CITI Technical Report 93–12

Adding 50,200 Entries to a DCE Namespace: A Comparison of OSF DCE V1.0.2 and IBM DCE/6000 V1.2

Mark R. Carter
markc@citi.umich.edu

ABSTRACT

This study compares two releases of the Distributed Computing Environment (DCE) during the addition of 50,200 server entries to the cell namespace. To conduct the study, we configured a one-machine cell with DCE core services (RPC facility, initial security server, primary CDS server, and DTS local server). The study focuses on the `cdsd` paging space, CDS file space usage, and `cdscp` response time. Although resource consumption increases linearly with the number of users added to the cell namespace, `cdscp` response time increases markedly during a CDS checkpoint operation. The CDS performance statistics included in this study enable the DCE administrator to establish a resource usage baseline. However, further investigation is required to ascertain whether CDS server replication will smooth the `cdscp` response time or exacerbate `cdscp` delays.

December 31, 1993

Center for Information Technology Integration
University of Michigan
519 West William Street
Ann Arbor, MI 48103-4943

Adding 50,200 Entries to a DCE Namespace: A Comparison of OSF DCE V1.0.2 and IBM DCE/6000 V1.2

Mark R. Carter

December 31, 1993

1. Introduction

At the Center for Information Technology Integration (CITI), we hypothesize that a large increase in the number of Remote Procedure Call (RPC) server entries in a Distributed Computing Environment (DCE) cell namespace will result in a linear increase in paging space usage and Cell Directory Service (CDS) database file sizes. We further hypothesize that CDS access times via `rpccp export` and `cdscp show` commands will remain fairly constant as the namespace grows. This study investigates these two hypotheses by comparing two versions of DCE: OSF DCE V1.0.2 (referred to as OSF) and IBM DCE/6000 V1.2 (referred to as IBM). The following issues, pertaining to the ability to scale a DCE cell namespace, are also addressed:

- *Feasibility*—Can an administrator add 50,200 entries to the namespace?
- *Time-Based Issues*—How long does it take to add 50,200 entries? Does it take longer to add an individual entry as the size of the cell namespace increases? Does it take longer to look up an entry as the size of the cell namespace increases?
- *Paging Space Issues*—How much does paging space usage increase when adding 50,200 entries? What is the paging space usage per individual entry during the add sequence? Does the paging space usage change after the add sequence is completed?
- *Database Size Issues*—How big do the DCE database files get when adding 50,200 entries? What is the file storage size per individual entry during the add sequence? Does the file storage size change after the add sequence is completed?

The CDS is not placed under any load except for that required to add the entries to the database. The `rpccp export` command is used to add entries to the namespace, and the `cdscp show` command is used to periodically check `cdscp` response time, but no other `cdscp` or `rpccp` commands are performed during the CDS tests. Test scripts and logs appear in the Appendices, and graphs of specific results appear in figures throughout the paper.

2. Method

The following sections outline the design and materials used in the test.

2.1 The Test Machine

The experiment ran on an IBM RS/6000 model 7013/530 machine with:

- AIX V3.2.4 operating system,
- AFS V3.2.2 distributed file system,
- 64 MB of physical RAM,
- One 857 MB hard drive, and
- Two paging spaces: `hd06` at 80 MB, and `Paging00` at 148 MB, for a total of 228 MB of paging space.

The size of the `/var` file system for the OSF (`/opt`) and IBM (`/var/dce`) DCE installation and configuration were as large as possible, typically around 250 MB.

2.2 DCE Cell Preparation

DCE test cells were configured immediately before starting the test suite execution. Cells are not used by others during testing.

To prevent network glitches from distorting the test results, all test cells were composed of only one server machine and the test suite was executed on the server machine. Only the following DCE core services were configured: `secd`, `sec_clientd`, `cdsd`, `cdsadv`, `cdsclerk`, and `dtsd` (running as a local server). The Global Directory Service (GDS) and the Distributed File System (DFS) were neither installed nor configured.

The default authorization policy for the DCE cell required some modification before beginning test suite execution. Namely, the “maximum certificate lifetime” setting was changed from 5 hours to 10 days and the “maximum certificate-renewable lifetime” setting was changed from 1 day to 10 weeks. Without this modification, the `dce_login` as `cell_admin` would expire before the conclusion of the test suite.

2.3 The Why and How of 50,200 Entries

CITI is exploring the feasibility of creating a single university-wide DCE cell that includes a large number of workstations—on the order of 50,000. The number 50,200 is roughly twice the length of the dictionary file (25,144) we used to create a set of unique test entry names.

Entry names in the namespace must be unique. To create a file with enough unique entries to test the predicted cell size of approximately 50,000, we started with an existing 25,144 line dictionary file (`usr/dict/words`). To create the balance of 50,200 entries, we duplicated the file, and prepended the character “z” to each word in the duplicate file. Then, the two files were concatenated, sorted uniquely, and trimmed to produce a file that contains over 50,200 entries. (Words that begin with a numeral were removed because this file will be used for additional DCE tests in which such words may be illegal. For the same reason, apostrophes in words were changed to underscores.)

The resulting file was divided into 502 files of 100 words each. Because the names are in alphabetical order, the entries are added to the namespace in alphabetical order. This is by design to ascertain whether this diabolical insertion order causes any problems. It does not.

2.4 An Entry

Adding an entry to the namespace means exporting an RPC server entry, including three object UUIDs, to the cell namespace.

2.5 The CDS Test Case

Our CDS test case exports 100 server entries to the namespace. Every group of 100 entries is created with the 100 corresponding server names obtained from the input file, and a CDS directory name derived from the first name in the file. Creating a new CDS directory for each batch of 100 entries more closely simulates actual conditions than does creating all 50,200 entries in one directory.

The test case accepts a list of words as input. Because the `rpccp` and `cdscp` command line interfaces are slow, each test case constructs a temporary file that contains `rpccp` and `cdscp` commands in the form of a `here` document. The file of commands is executed as a shell program.

The *case time* is the time required to execute one test case. Case times are reported in seconds.

2.6 The CDS Test Suite

There are 502 test cases in the test suite. The CDS test suite adds 50,200 entries to the namespace.

The status of the DCE environment is checked between each test case rather than at regular intervals to compute the time required to add the entries from the time stamp in the status report, and to determine the number of entries in the namespace when the status is obtained.

While the test suite is running, another process monitors `cdscp` response time with a `cdscp show cell /.:/` command once per minute.

The *check time* is the time required to execute one `cdscp show cell /.:/` command. Check times are reported in seconds.

3. Procedure

The following sections outline the procedure used to implement the text.

3.1 Namespace Test Suite Execution

The test case script generates a file that contains `rpccp` and `cdscp` commands, and this file is executed by the test script. No `rpccp` or `cdscp` commands are executed directly in the test case script.

Please see Appendix A for a listing of the `cds_test_driver` script and Appendix B for a listing of the `cds_test_case` script. Please see Appendix C for an abbreviated listing of a typical test case temporary file.

Two log files are obtained from each test suite, the `testlog` and the `checklog`. Pertinent information is extracted from the log files and put into tabular form. Case times are computed and summary statistics are obtained from the tables. The data from some tables are graphed.

3.2 Monitoring the Test Cases—DCE Environment Status

Between each test case the status of the DCE environment is recorded. Several short shell scripts are used to ascertain all but the first item which the driver script produces:

- Test case number (from 1 to 502)
- The date (day of month) and time (hh:mm:ss)
- `vmstat` command
- Paging space (percent used via `lsps -a`)
- Memory used by DCE core processes (via `ps xv` and `grep`)
- DCE core database file sizes (via `ls -l`)

See Appendix D for a complete listing of the scripts used to obtain status information. A partial DCE environment status report appears in Appendix E. These status reports, along with anything the `rpccp` or `cdscp` program prints, are collected in the `testlog` file using the `script` command.

The size of the `testlog` files is typically about 2.4 MB and they contain output captured from the screen during test suite execution. This output consists of a mixture of the status report and `rpccp` and `cdscp` feedback. See Appendix F for a partial listing of an example `testlog`.

3.3 Monitoring the Test Cases—CheckCDS

The purpose of the `CheckCDS` script is to monitor the response time of the CDS control program, `cdscp`. The script is executed in the background, and is automatically started by the `cds_test_driver` script. See Appendix G for a listing of the `CheckCDS` script.

At regular intervals, the `CheckCDS` script issues a `cdscp show cell /.:/` command. The output from the `cdscp` command, and the execution time are recorded in the `checklog` file by the `CheckCDS` script. If the `cdscp` command takes longer than the specified interval to return, an error message is also recorded in the `checklog` file. Note, the check interval is held constant by adjusting the `sleep` duration. The `sleep` time is calculated by subtracting the desired check interval time from each `cdscp` command execution time. This results in a more regular sample interval than does a simple `sleep N` because the duration of the `cdscp` command itself is considered.

The size of a `checklog` file is typically about 225 KB. See Appendix H for a partial listing of an example `checklog`.

4. Results

The test suite completed successfully all test cases.

After the test suite execution, we queried the namespace to get a count of CDS directories and objects. The count totals were as expected. In other words, all CDS directories and RPC entries that the test suite adds to the namespace actually appear in the namespace. Table 1 shows the number of items in the namespace at the conclusion of the test suite. Namespace entries added during cell configuration are not included in this count.

TABLE 1. Items in the Namespace after Test Suite Execution

Number of Items in the Namespace	OSF	IBM
CDS directories in `/.:/hosts/test/`	502	502
Objects in every `/.:/hosts/tests/*` directory	100	100
Number of CDS directories that DO NOT contain 100 objects	0	0
Number of CDS objects added to the namespace	50,200	50,200

The following sections outline specific results of the test.

4.1 Time to Execute the Complete Test Suite

The execution time of the CDS test suite totaled 38.4 hours for OSF and 39.3 hours for IBM.

The overall average time for adding one entry to the namespace was 2.75 seconds for OSF and 2.82 seconds for IBM. The graph in Figure A shows the serial number of the test case (OSF and IBM) across the elapsed time.

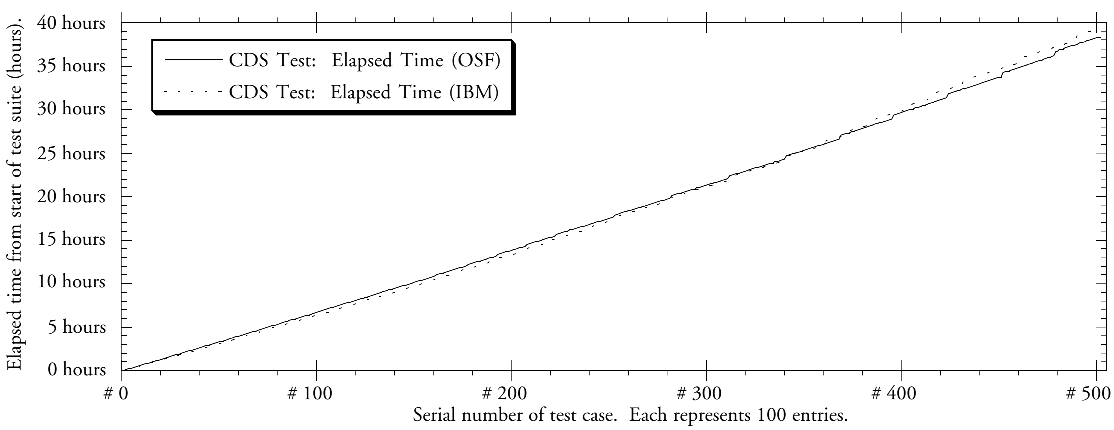


Figure A. Test Suite Execution Time

4.2 Case Times

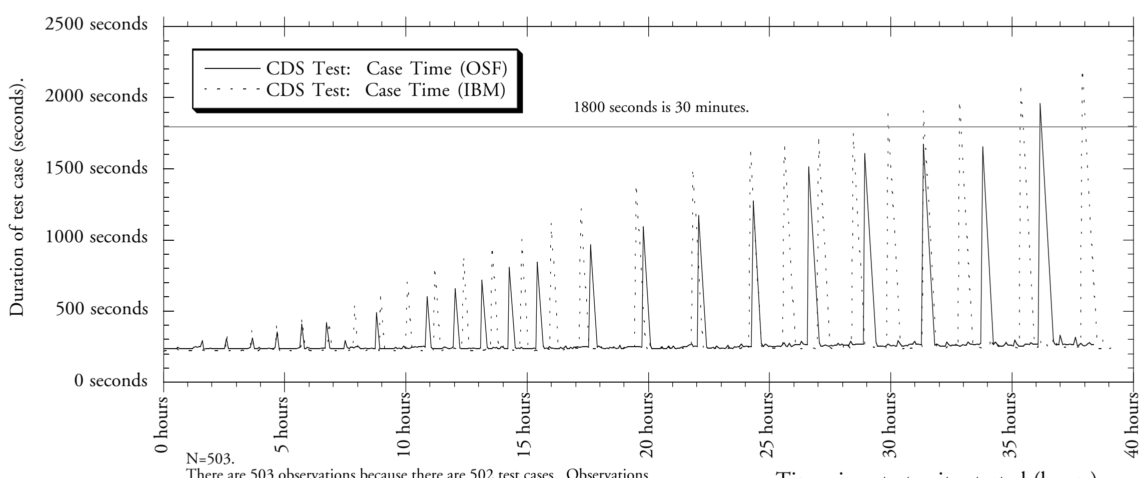
The overall average case time was 276 seconds for OSF and 279 seconds for IBM.

On average, test cases execute slightly faster on the OSF release. Case times increased slightly as the size of the namespace increased (Table 2).

TABLE 2. Average Case Time as Size of Registry Increases

Registry size	Average OSF case time in seconds	Average IBM case time in seconds
1-50	240	225
51-100	245	235
101-150	249	241
151-200	266	261
201-250	265	277
251-300	275	285
301-350	289	295
351-400	314	332
401-450	291	348
451-500	329	286

However, the maximum case time was 1959 seconds for OSF and 2102 seconds for IBM. Figure B shows CDS test case times (OSF and IBM) graphed against the elapsed time from the beginning of test suite.



N=503.
There are 503 observations because there are 502 test cases. Observations are obtained before the first test case, and following each subsequent test case.

Figure B. Case Times

Occasional lengthy case times (> 400 seconds) occur in both releases. The lengthy times cluster around CDS checkpoints—when `cdsd` saves its entire database from virtual memory to disk. Although the average time required to complete a test case is not significantly greater because of the checkpoint delays, the maximum time for a test case is increased. Tables 3 and 4 show the average and maximum case times (OSF and IBM), excluding and including checkpoint slowdowns.

TABLE 3. Average Case Times

	Average OSF case time in seconds	Average IBM case time in seconds
excluding slowdowns	245	237
including slowdowns	276	279

TABLE 4. Maximum Case Times

	Maximum OSF case time in seconds	Maximum IBM case time in seconds
excluding slowdowns	317	332
including slowdowns	1,959	2,102

Tables 5 and 6 show frequency distributions of case times during testing on the OSF release. Of the 502 total observations, 484 test cases (97%) were completed in less than 401 seconds and only 17 (3%) took longer than 402 seconds to complete. Moreover, 467 test cases (93%) completed between 220 and 280 seconds.

Table 5. Frequency Distribution of the Time (in seconds) Required to Complete One Test Case (OSF)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	1	201	0	0%	
2	201	401	484	96.6%	- Mode
3	401	601	4	.8%	
4	601	801	2	.4%	
5	801	1001	3	.6%	
6	1001	1201	2	.4%	
7	1201	1401	1	.2%	
8	1401	1601	1	.2%	
9	1601	1801	3	.6%	
10	1801	2001	1	.2%	
11	2001	2201	0	0%	
12	2201	2401	0	0%	

Table 6. Frequency Distribution of the Time (in seconds) Required to Complete Cases Between 200 and 400 seconds (OSF)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	200	220	0	0%	
2	220	240	184	36.7%	
3	240	260	209	41.7%	- Mode
4	260	280	74	14.8%	
5	280	300	12	2.4%	
6	300	320	3	.6%	
7	320	340	1	.2%	
8	340	360	1	.2%	
9	360	380	0	0%	
10	380	400	0	0%	

Tables 7 and 8 show frequency distributions of case times during testing on the IBM release. Of the 502 total observations, 478 test cases (95%) were completed in less than 201 seconds and only 24 (5%) took longer than 401 seconds to complete. Moreover, 457 test cases (91%) were completed between 200 and 260 seconds.

Table 7. Frequency Distribution of the Time (in seconds) Required to Complete One Test Case (IBM)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	1	201	0	0%	
2	201	401	478	95.4%	- Mode
3	401	601	4	.8%	
4	601	801	2	.4%	
5	801	1001	3	.6%	
6	1001	1201	2	.4%	
7	1201	1401	2	.4%	
8	1401	1601	1	.2%	
9	1601	1801	4	.8%	
10	1801	2001	3	.6%	
11	2001	2201	2	.4%	
12	2201	2401	0	0%	

Table 8. Frequency Distribution of the Time (in seconds) Required to Complete Cases Between 200 and 400 seconds (IBM)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	200	220	20	4%	
2	220	240	276	55.1%	- Mode
3	240	260	161	32.1%	
4	260	280	11	2.2%	
5	280	300	4	.8%	
6	300	320	4	.8%	
7	320	340	1	.2%	
8	340	360	0	0%	
9	360	380	1	.2%	
10	380	400	0	0%	

The checkpoint operation takes longer as the size of the namespace increases. Therefore, the larger the size of the namespace, the more likely it is that a test case will execute during a checkpoint. Several test cases often execute during a single checkpoint. It is clear that individual `rpccp export` and `cdscp show` commands do not block because of the checkpoint. Instead, the execution time of both commands increases.

4.3 Paging Space Usage

Paging space consumption increases linearly with the number of entries added to the namespace, but decreases considerably when DCE services are restarted at the end of the test suite.

Paging space usage increases 109 MB for OSF and 108 MB for IBM. The paging space growth was 2175 bytes per entry (OSF) and 2159 bytes per entry (IBM). Paging space usage was similar in both test conditions (OSF and IBM). Also in both test conditions, the maximum amount of paging space usage occurred immediately before the end of the test suite. Paging space usage approached approximately 60% of available space. Table 9 shows a summary of paging space usage.

TABLE 9. Paging Space Usage Summary Per Entry

Paging space usage summary	OSF	IBM
Number of entries added to the namespace (count)	50,200	50,200
Paging space usage at start of add sequence (MB)	31.72	34.80
Paging space usage at end of add sequence (MB)	140.88	143.16
Growth in paging space used (MB)	109.16	108.36
Growth rate (Bytes/user)	2175	2159

Figure C shows the total paging space used (OSF and IBM) across the elapsed time from the beginning of test suite.

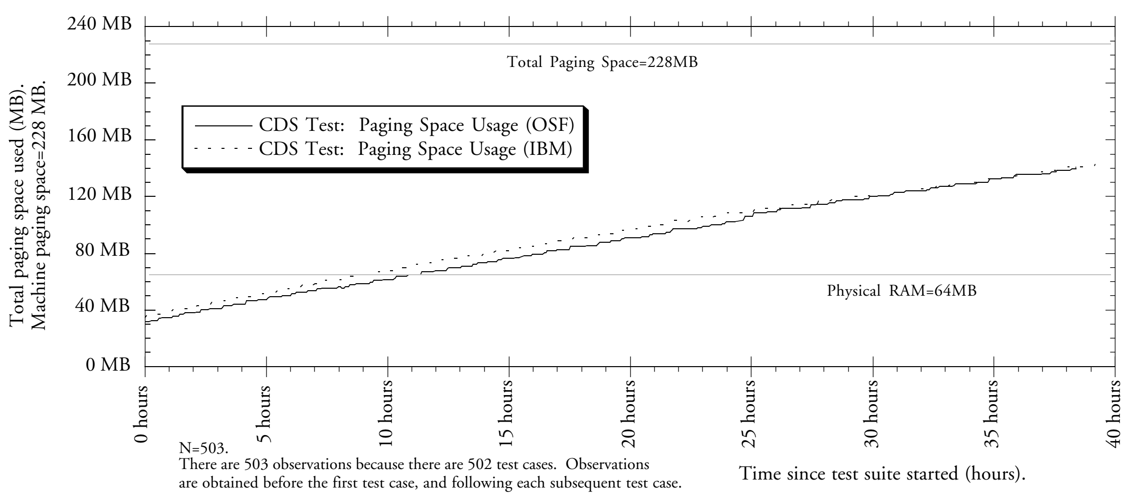


Figure C. Paging Space

4.4 Paging Space After the Test Suite

After all entries were added to the namespace, DCE core processes were halted and restarted. The paging space usage was measured immediately after DCE was restarted. Table 10 shows a summary of this paging space usage.

TABLE 10. Estimate of the Amount of Paging Space Required for Each User

Paging space usage after DCE restart	OSF	IBM
Size of paging space before DCE started (MB)	31.72	34.8
Size of paging space after DCE restart (MB)	110.88	N/A
Difference from starting size.	79.16	N/A
Paging space usage after restart (Bytes/user)	1577	N/A

As entries are added to the namespace, the average amount of paging space per user is 2175 bytes for OSF and 2159 bytes for IBM. However, after all entries are added and DCE is restarted, the average amount of paging space per user drops to 1577 bytes (OSF). Services were not restarted, nor paging space measured on the IBM release. We believe the results would be similar on the IBM release.

4.5 Size of DCE Database Files

The combined size of the CDS database files increases linearly with the size of the namespace. The size of other DCE database files remains almost constant as entries are added to the namespace.

The size of the CDS database files increased 69 MB for OSF and 103 MB for IBM. The CDS database files are `cds_files`, `server_mgmt_acl.dat`, `*checkpoint*`, `*tlog*`, and `*version`, and they all reside in `/opt/dcelocal/var/security/rgy_data`. The size of the registry data base file increased 1378 bytes per user (OSF), and 2054 bytes per user (IBM). The increase in size of the CDS files is considerably greater on the IBM release.

Tables 11 and 12 show the minimum and maximum combined size of all DCE database files and CDS database files respectively.

TABLE 11. DCE Database File Size Summary

DCE database file size summary	OSF	IBM
Size of DB files at start of test suite (bytes)	206,321	212,629
Size of DB files at end of test suite (bytes)	79,195,980	109,228,976
Growth of DB files (bytes)	78,989,659	109,016,347
Size of DB files per user (bytes/user)	1,573	2,172

TABLE 12. CDS Database File Size Summary

DCE CDS database file size summary:	OSF	IBM
Size of CDS DB at start of test suite (bytes)	4,969	4,988
Size of CDS DB at end of test suite (bytes)	69,198,185	103,126,908
Growth of CDS DB (bytes)	69,193,216	103,121,920
Size of CDS DB per user (bytes/user)	1,378	2,054

When sizing the DCE `/var` file system, the administrator must consider the large, but transient increases in file space usage during CDS checkpoints. For example, on the IBM release, the next checkpoint will use an extra 103+ MB to create the next checkpoint file, before removing the old checkpoint file and reducing the size of the transaction log. (See Appendix I for an in depth explanation.)

The graph in Figure D shows the combined DCE core services database files size across the elapsed time from the beginning of the test suite (OSF and IBM).

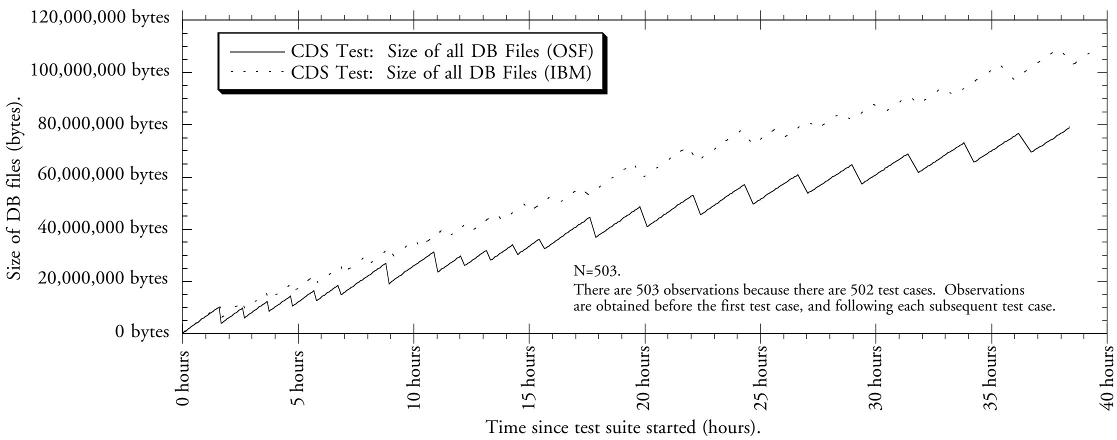


Figure D. DCE Database File Sizes

The graph in Figure E shows the CDS combined database files (except for the transaction log file) and the transaction log file sizes across the elapsed time from the beginning of the test suite (IBM). The data for the OSF case is similar. Note that the CDS database files grow when the transaction file is emptied during a checkpoint.

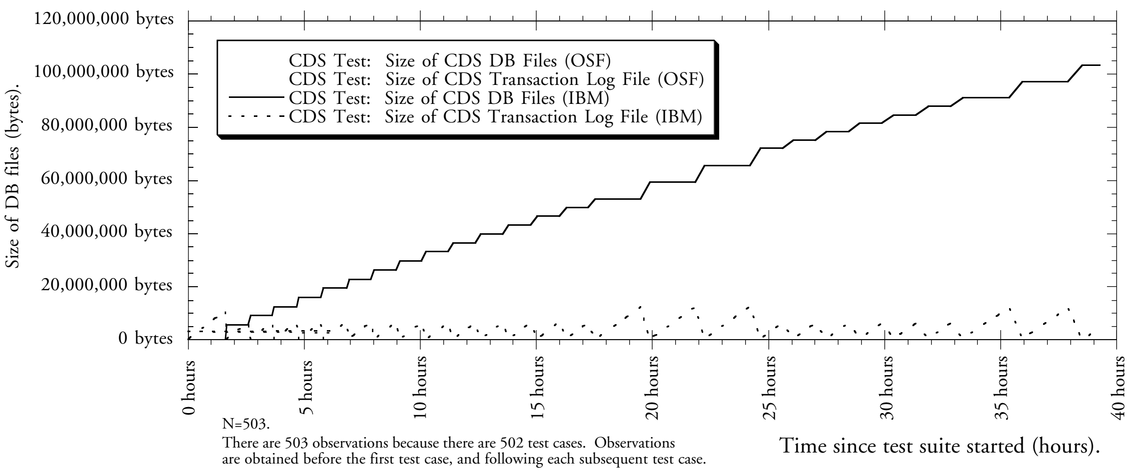


Figure E. CDS Database File Sizes

In both test conditions (OSF and IBM), the minimum size of the namespace database files occurs at the beginning of the test suite, and the maximum size of the files occurs at the end of the test suite. Restarting DCE services does not affect the size of the files.

4.6 Check Times

During CDS test suite execution, `CheckCDS` initiates a `cdscp show cell /.:/` command approximately once every minute. The time required to execute this command is called the check time. The time required to view a CDS object remains virtually constant as the size of the namespace increases—unless the CDS is checkpointing. Check times were not obtained during the OSF condition. We believe we would observe similar results on the OSF release.

Although the average check time is 16.1 seconds, the median check time is 3.0 seconds. This suggests there are occasional lengthy check times. If the CDS is engaged in a checkpoint operation, the check time increases considerably. Table 13 shows summary statistics on the check times. Tables 14 and 15 show a frequency distribution of the check times.

TABLE 13. Summary Statistics of the Check Time in Seconds (IBM)

Check Time	CDS
Count (N=)	1994
Minimum time (seconds)	1
Maximum time (seconds)	1031
Average time (seconds)	16.1
Median time (seconds)	3

Of the 1994 total commands observed, 1885 (96%) returned in less than 68 seconds, and only 51 (3%) took longer than 68 seconds. Moreover, 1504 commands (75%) returned in 5 seconds or less, and 1071 (54%) returned in 3 seconds or less.

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	0	68.7	1942	97.4%	- Mode
2	68.7	137.5	9	.5%	
3	137.5	206.2	2	.1%	
4	206.2	274.9	4	.2%	
5	274.9	343.7	2	.1%	
6	343.7	412.4	4	.2%	
7	412.4	481.1	6	.3%	
8	481.1	549.9	4	.2%	
9	549.9	618.6	9	.5%	
10	618.6	687.3	5	.3%	
11	687.3	756.1	3	.2%	
12	756.1	824.8	1	.1%	
13	824.8	893.5	1	.1%	
14	893.5	962.3	1	.1%	
15	962.3	1031	1	.1%	

Table 14. Frequency Distribution of the Check Time in Seconds (IBM)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	1	3	1071	53.7%	- Mode
2	3	5	433	21.7%	
3	5	7	162	8.1%	
4	7	9	139	7%	
5	9	11	80	4%	
6	11	13	8	.4%	

Table 15. Frequency Distribution of Check Times Less than 13 Seconds (IBM)

Although check times are usually less than 9 seconds, some are significantly longer—up to 1,031 seconds. Figure F graphs the registry test check times against elapsed time from the beginning of the test suite (IBM only). Figure G plots those check times under 20 seconds, showing clustering under 15 seconds (IBM only).

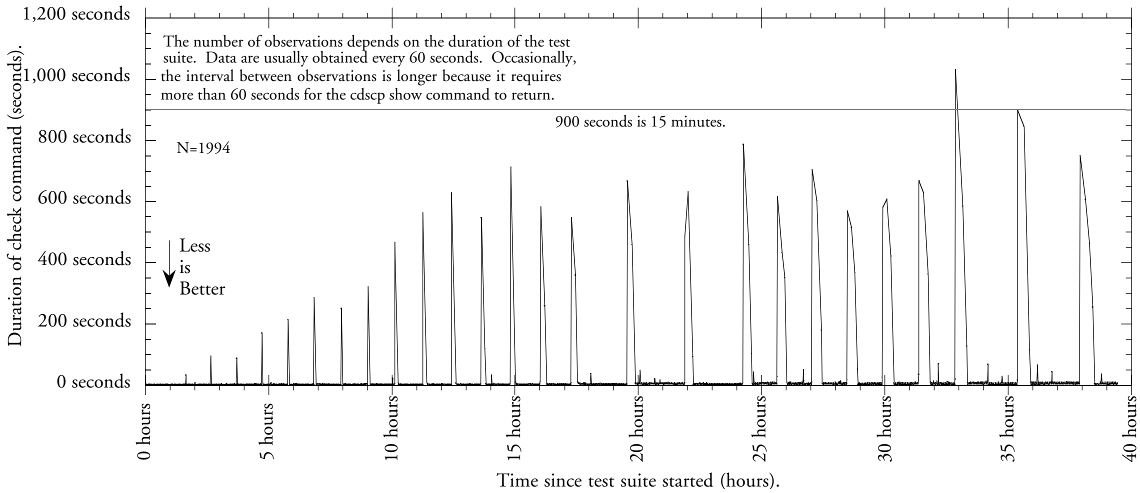


Figure F. Check Times

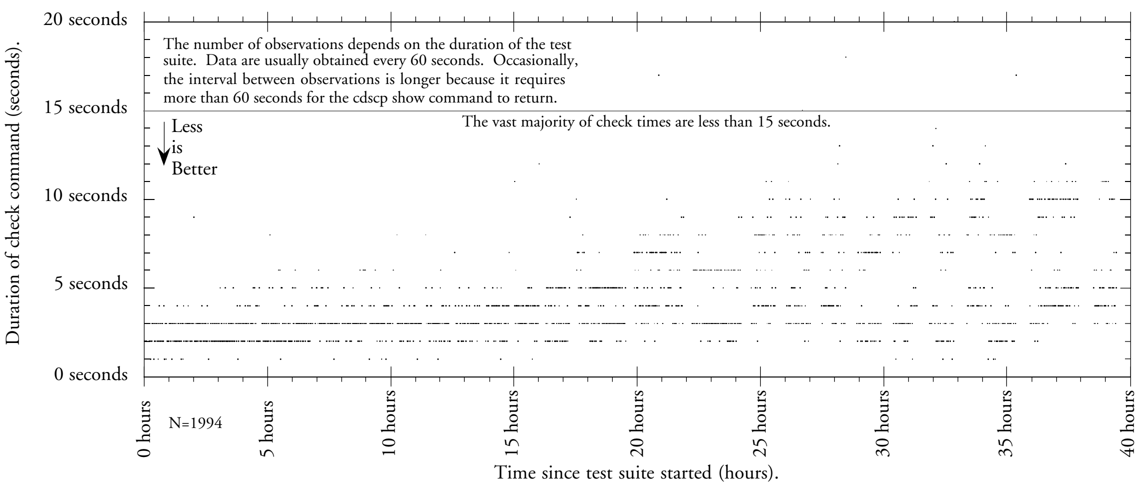


Figure G. Check Times Under 20 Seconds

Longer check times cluster around checkpoints, and occur more frequently as the namespace size increases. Although the majority of check times are less than 20 seconds, Figure G reveals a trend toward slightly longer times.

4.7 Why the Delays?

Case times and check time are substantially greater when the CDS is checkpointing.

Please see the graph in Figure H, which shows the percent of maximum value across the elapsed time from the beginning of the test suite (IBM). This graph demonstrates the case time and check time delays. Check times were not obtained during testing on the OSF release, but we believe we would observe similar results.

The *Percent of Maximum Value* is a derived unit that represents the proportion of an observed statistic to the maximum value that statistic reached during the test. In this manner, one can easily compare two observed statistics, for example `cdscp` response time and the size of the CDS transaction log, even though the absolute magnitude of the values of each statistic ranges over many orders of magnitude.

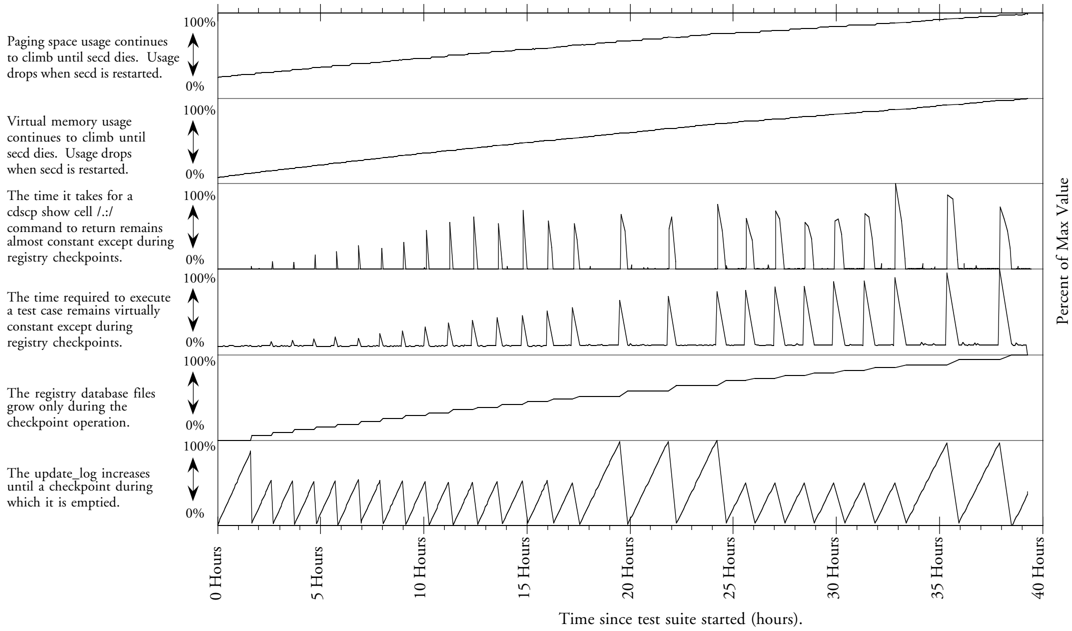


Figure H. Maximum Values (IBM)

This graph contains six layers, one upon the next. Layer 1 is on the bottom. Layer 1 (saw tooth) represents the size of the CDS transaction log file. Layer 2 (stair step) represents the combined size of all the CDS database files except the transaction log. When the CDS checkpoint file size increases, the transaction log size decreases. This is evidence of a CDS checkpoint. Notice that some CDS checkpoints seem to occur at twice the usual interval.

Layer 3 represents case times. Layer 4 represents check times. Longer than usual times are demonstrated by the spikes that occur while the CDS is checkpointing.

Layers 5 and 6 represent DCE virtual memory usage and machine paging space, respectively. The consumption of these two resources rises steadily as the test suite executes.

4.8 Checkpoint Frequency During the Test

CDS checkpoints are frequent when the CDS is heavily exercised.

The CDS test suite adds 50,200 entries to the namespace as quickly as possible. During testing, the CDS checkpointed 21 times (OSF) and 26 times (IBM). On average, checkpoints occurred every 1.75 hours (OSF) and 1.47 hours (IBM). However, the interval between checkpoints is strongly bimodal and the average intercheckpoint time is a poor summary indicator. The graph in Figure I, shows CDS intercheckpoint times graphed against the checkpoint number (OSF and IBM).

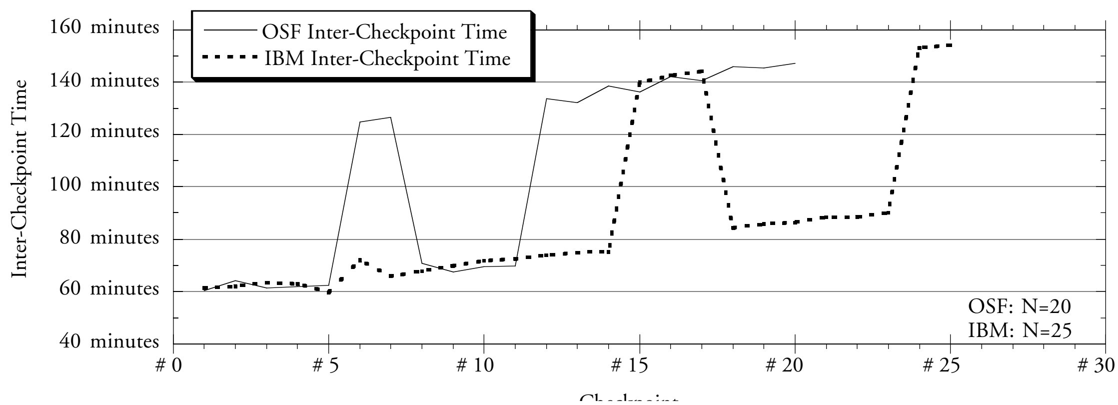


Figure I. CDS Intercheckpoint Times

On average, checkpoints occurred after the addition of 2265 entries (OSF) and 1848 entries (IBM). Figure J shows the CDS intercheckpoint entry counts graphed against the checkpoint number. In this case, the entries are server entries and are rounded to the nearest 100.

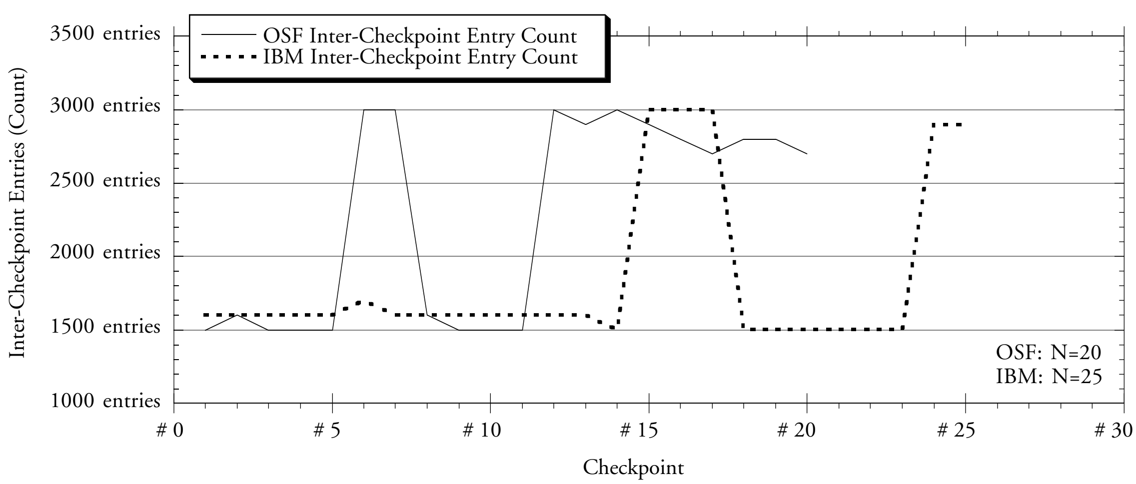


Figure J. CDS Intercheckpoint Entry Counts

Interchekpoint times are similar on the OSF and IBM releases, and tend to increase as the size of the namespace increases. This may happen because `cdsd` does not include the time required for the checkpoint operation in the calculation of when to checkpoint. For unknown reasons, the inter-checkpoint times cluster around two values.

Intercheckpoint entry (server entry) counts are only slightly different on the two releases, and the counts remain steady as the size of the namespace increases. Intercheckpoint entry counts cluster around two values that depend on the intercheckpoint time.

From these data, it is difficult to ascertain exactly what triggers a CDS checkpoint.

4.9 Checkpoint Frequency—In an Idle Cell

Checkpoints are infrequent when the cell is not in use.

Check times were observed for six days while the namespace was not changed. Table 16 shows a frequency distribution of the check times (in seconds) in an idle cell.

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	0	4	3956	94.4%	- Mode
2	4	8	171	4.1%	
3	8	12	7	.2%	
4	12	16	12	.3%	
5	16	20	19	.5%	
6	20	24	7	.2%	
7	24	28	6	.1%	
8	28	32	5	.1%	
9	32	36	5	.1%	
10	36	40	1	2.4E-2%	
11	40	44	1	2.4E-2%	

Table 16. Frequency Analysis of Check Times in an "Idle" Cell Over a Six Day Period

Of the 4,190 observations, most (95%) are 4 seconds or less, and many (98%) are 8 seconds or less. No check times are longer than 44 seconds. The CDS checkpointed several times during this observation—but the dates and sizes of the CDS files remained unchanged.

5. Conclusion

It requires 38.4 hours for OSF and 39.3 hours for IBM, to add 50,200 entries to a DCE cell namespace.

Namespace size does not affect case times (the time required to add 100 entries to the namespace) unless the CDS is checkpointing. Excluding checkpoints, the average case time is 245 seconds (OSF) and 237 seconds (IBM). The maximum case time is 317 seconds (OSF) and 332 seconds (IBM). CDS checkpoints diminish `rpccp export` performance significantly. Including checkpoints, the average case time is 276 seconds (OSF) and 279 seconds (IBM). The maximum case time increases to 1,959 seconds (OSF) and 2,102 seconds (IBM).

Namespace size does not affect the check times (the time required for a `cdscp show cell /.:/` to return) unless the CDS is checkpointing. The median check time is 3.0 seconds. Registry checkpoints increase check times significantly. The average check time is 16.1 seconds, and the maximum check time is 1,031 seconds. Check times were obtained only on the IBM release. We believe we would observe similar results on the OSF release.

CDS checkpoints (when `cdsd` saves its entire database from virtual memory to disk) are frequent when the CDS is under heavy load. On average, checkpoints occur every 1.75 hours (OSF) and 1.47 hours (IBM). During testing, the CDS checkpointed 21 times (OSF) and 26 times (IBM).

However, CDS checkpoints are infrequent when the cell is not in use. When no changes were made to a cell namespace for six days, few CDS checkpoints occurred. DCE resource usage (machine paging space, DCE processes virtual memory consumption, and database file sizes) does not change significantly when the cell is idle.

Paging space usage increases linearly as entries are added to the namespace. The paging space usage increased 109 MB (OSF) and 108 MB (IBM). The paging space usage per individual user reached 2175 bytes per user (OSF) and 2159 bytes per user (IBM), during the add sequence. The paging space usage drops to 1577 bytes per entry (OSF) when DCE daemons are restarted after the add sequence is completed.

During the test, the CDS database files grew approximately 70 MB (OSF) and 103 MB (IBM), yet the `SIZE` of the `secd` process and the paging space usage increased approximately 109 MB (OSF and IBM). When `cdsd` was restarted after the test, paging space consumption and virtual memory usage was considerably reduced, yet no namespace data were lost. This suggests an underlying problem with the `cdsd` code: Why does usage drop when the daemons are restarted? Why do database files grow much larger on the IBM release than on the OSF release?

The size of most DCE database files remains virtually constant as the size of the namespace increased. The size of the CDS database files increased linearly as the size of the namespace increased. The size of the transaction log increased until a checkpoint, and then returned to a baseline size. The size of all database files increased 79 MB (OSF) and 109 MB (IBM). The CDS database files account for most of the increase. The file space usage per individual user during the add sequence is 1378 bytes (OSF) and 2054bytes (IBM). The file space usage does not change when DCE daemons are restarted after the add sequence is completed.

Although one can successfully add 50,200 entries to a single DCE cell namespace, one can expect linear resource consumption as a result. Because the CDS server maintains a working copy of the namespace database in virtual memory, `cdsd` may reach an AIX limit on the `SIZE` of its data segment and crash. The need to maintain the entire namespace in core memory questions the viability of the CDS in a large-scale cell. Delays associated with CDS checkpoints increase response time while the entries are being added, and periodically thereafter. Further investigation is required to ascertain whether CDS server replication will smooth the `cdscp` response time or exacerbate `cdscp` delays. That a read-only copy of the namespace data is made available by replication suggests improvement in `cdscp` lookup time during checkpointing. However, `rpccp exports` may remain slow because they are addressed to the master (writable) replica. Moreover, the increased overhead associated with CDS replication may offset any possible improvement.

Acknowledgments

This work was supported by IBM.

AppendixA: `cds_test-driver` Script

```
#!/bin/ksh
write_log()

{
/DCESuitcase/test/log_date
vmstat
/DCESuitcase/test/check_swap_space
/DCESuitcase/test/check_dce_memory
/DCESuitcase/test/check_dce_files
}

cds_test_suite()
{
MAX_NUMBER_OF_CASES=502
CASE_NUMBER=1

let ENTRIES_ADDED=MAX_NUMBER_OF_CASES-CASE_NUMBER+1
let ENTRIES_ADDED=ENTRIES_ADDED\*100
echo "

You will run cases #$CASE_NUMBER to #$MAX_NUMBER_OF_CASES which will add a total of $EN-
TRIES_ADDED entries to the CDS database.

"

cdscp list directory /.:/hosts/test | egrep -v "LIST|DIRECTORY|AT" | grep test > /dev/null
if [ $? -ne 0 ]
then
        echo "Creating CDS directory /.:/hosts/test"
        cdscp create directory /.:/hosts/test
else
        echo "CDS directory /.:/hosts/test exists.  That's OK"
fi

while [ $CASE_NUMBER -le $MAX_NUMBER_OF_CASES ]
do
        if [ $CASE_NUMBER -gt 99 ]
        then
                FILE=../NAMES/names.$CASE_NUMBER
        elif [ $CASE_NUMBER -gt 9 ]
        then
                FILE=../NAMES/names.0$CASE_NUMBER
        else
                FILE=../NAMES/names.00$CASE_NUMBER
        fi
        echo "cds_test_driver: Running CASE $CASE_NUMBER"
        write_log
        cat $FILE | cds_test_case
        CASE_NUMBER=`expr $CASE_NUMBER + 1`
done
write_log
}

prelim_check()
{
clear
    echo "
```

```
You must have already started the script program to capture the output from this test suite.

Press Return/Enter to continue, or Control-C to stop.

"
read a
    echo "Checking to be sure you are dce_login'd as cell_admin."
    result=`/bin/klist 2>/dev/null | grep "Principal:" | grep -v "Global" | \
            awk '{print $3}'`
    result=${result:-"NO_DCE_LOGIN"}
    if [ $result != "cell_admin" ]
    then
        echo "

  Oops, you are dce_login'd as: \"$result\"

  and you haven't dce_login'd as cell_admin.

  Exiting this test driver so you can...

"
    exit 1
    fi
}

prelim_check; # Be sure things are set for testing.

if [ -f CheckCDS ]
then
        CheckCDS -force 60 /tmp/cds.check.log &
        sleep 4
else
        echo "Can't find CheckCDS, you may want to start it by hand."
        echo "\n\nPress Return to continue...\n\n"
        read a
fi

cds_test_suite; # Run the test suite.

sleep 100
kill -9 `ps xv | grep CheckCDS | grep -v grep | awk '{print $1}'`

exit 0
```

Appendix B: `cds_test_case` Script

```
#!/bin/ksh

TEMPFILE=/tmp/cds.test.$$

echo "cds_test_case: Output file is $TEMPFILE"
exec 3>&1 > $TEMPFILE
(( index=-1 ))
while read a
do
        (( index=$index+1 ))
        Names[$index]=$a
done
(( NUMBER=$index ))

echo "#!/bin/sh"

DIR=${Names[0]}
echo "cdscp create directory /.:/hosts/test/$DIR"

echo "rpccp <<EOF"

(( index=-1 ))
while (( index < $NUMBER ))
do
        (( index=$index+1 ))
        # Create/Export ficticious servers
        echo "export -i `uuidgen` -b ncadg_ip_udp: -b ncacn_ip_tcp: -o `uuidgen` \
-o `uuidgen` -o `uuidgen` \
-s dce /.:/hosts/test/$DIR/${Names[$index]}"
done

echo "quit
EOF"

exec 1>&3 3>&-

chmod +x $TEMPFILE
$TEMPFILE

/bin/rm $TEMPFILE
```

Appendix C: Partial Listing of a Typical Test Case Temporary File

```
#!/bin/sh
cdscp create directory /.:/hosts/test/worrisome
rpccp <<EOF
export -i 008ec4b6-365a-1cfa-af7a-02608c2c83b2 -b ncadg_ip_udp: -b ncacn_ip_tcp:
 -o 000a080c-365b-1cfa-a39d-02608c2c83b2 -o 001cc0e6-365b-1cfa-9bc0-02608c2c83b2
 -o 0030c8d4-365b-1cfa-b2e7-02608c2c83b2 -s dce /.:/hosts/test/worrisome/worrisome
export -i 0043cc40-365b-1cfa-9888-02608c2c83b2 -b ncadg_ip_udp: -b ncacn_ip_tcp:
 -o 0056507c-365b-1cfa-989c-02608c2c83b2 -o 0068dbf2-365b-1cfa-8b61-02608c2c83b2
 -o 007bcb90-365b-1cfa-af5d-02608c2c83b2 -s dce /.:/hosts/test/worrisome/worry
export -i 008ea4ae-365b-1cfa-b56a-02608c2c83b2 -b ncadg_ip_udp: -b ncacn_ip_tcp:
 -o 0009f27c-365c-1cfa-856f-02608c2c83b2 -o 001c9b98-365c-1cfa-a010-02608c2c83b2
 -o 002fae90-365c-1cfa-92c9-02608c2c83b2 -s dce /.:/hosts/test/worrisome/worse
export -i 0042915e-365c-1cfa-9141-02608c2c83b2 -b ncadg_ip_udp: -b ncacn_ip_tcp:
 -o 0054ffc4-365c-1cfa-abc0-02608c2c83b2 -o 00678540-365c-1cfa-b582-02608c2c83b2
 -o 007a8262-365c-1cfa-86d2-02608c2c83b2 -s dce /.:/hosts/test/worrisome/worsen
•••
        { 92 export statement omitted. }
•••
export -i 00081bbe-368e-1cfa-a51b-02608c2c83b2 -b ncadg_ip_udp: -b ncacn_ip_tcp:
 -o 001ab094-368e-1cfa-93a6-02608c2c83b2 -o 002dbff4-368e-1cfa-936b-02608c2c83b2
 -o 00403e40-368e-1cfa-9ad5-02608c2c83b2 -s dce /.:/hosts/test/worrisome/yen
export -i 0053ba56-368e-1cfa-a857-02608c2c83b2 -b ncadg_ip_udp: -b ncacn_ip_tcp:
 -o 006645f4-368e-1cfa-aa03-02608c2c83b2 -o 0078d296-368e-1cfa-9894-02608c2c83b2
 -o 008bddfa-368e-1cfa-8469-02608c2c83b2 -s dce /.:/hosts/test/worrisome/yeoman
export -i 00062246-368f-1cfa-9652-02608c2c83b2 -b ncadg_ip_udp: -b ncacn_ip_tcp:
 -o 0018a97a-368f-1cfa-a614-02608c2c83b2 -o 002bae6c-368f-1cfa-a3af-02608c2c83b2
 -o 003e22b8-368f-1cfa-8a2b-02608c2c83b2 -s dce /.:/hosts/test/worrisome/yeomanry
export -i 0051a2d4-368f-1cfa-82a5-02608c2c83b2 -b ncadg_ip_udp: -b ncacn_ip_tcp:
 -o 00643b74-368f-1cfa-9d1e-02608c2c83b2 -o 0076a8a4-368f-1cfa-8c42-02608c2c83b2
 -o 008922fe-368f-1cfa-b946-02608c2c83b2 -s dce /.:/hosts/test/worrisome/yeshiva
quit
EOF
```

Appendix D: Listing of Scripts Used to Obtain Status Information

/DCESuitcase/test/log_date

```
#!/bin/sh

#==============================================================================
#
# This script prints a standard date.  The format is:
#
#    DATE: date hour minute second.
#
# all separated by tabs.
#
# mrc 8 November 1993
#
#==============================================================================

echo -n "DATE:    "
date "+%d    %H    %M    %S"
```

/DCESuitcase/test/check_swap_space

```
#!/bin/sh

#==============================================================================
#
# This script prints the paging space usage on a machine.
#
# Output:
#
# N     name
#
# Where N is the percent used (you must know the MB to convert) and
# name is the name of the paging space.
#
# mrc  8 November 1993
#
#==============================================================================

lsps -a | grep -v "%Used" | awk '{printf ("%s\t%s\n", $5, $1)}'
```

```
/DCESuitcase/test/check_dce_files

#!/bin/sh

CMD=`basename $0`

show_rpc()
{

    RPCDIR=/opt/dcelocal/var/rpc
    if [ -d $RPCDIR ]
    then
        echo "RPC DB files in $RPCDIR"
        ls -l $RPCDIR
    else
        echo "$RPCDIR does not exist.\n  Is this a DCE machine?"
    fi
}

show_sec()
{
    SECDIR=/opt/dcelocal/var/security/rgy_data
    if [ -d $SECDIR ]
    then
        echo "Security DB files in $SECDIR"
        ls -l $SECDIR
    else
        echo "$SECDIR does not exist.\n  Is this a DCE security server machine?"
    fi

        ls -l /krb5/*
}

show_cds()
{
    CDSDIR=/opt/dcelocal/var/directory/cds
    if [ -d $CDSDIR ]
    then
        echo "CDS DB files in $CDSDIR"
        ls -l $CDSDIR
    else
        echo "$CDSDIR does not exist.\n  Is this a DCE CDS server machine?"
    fi
}


show_rpc
show_sec
show_cds
```

```
/DCESuitcase/test/check_dce_memory

#!/bin/sh

#===========================================================================
#
# This command reports the memory used by DCE processes in (on RS/6K AIX3.2)
# the sixth column.
#
# This command will show all DCE processes with no command line arguments,
# or, only those DCE processes listed in an "egrep" style string:
#
#   check_dce_memory "rpcd|cdsd"
#
# will report only rpcd and cdsd.  The quotes are required.
#
# MRC 13 July 1993; 8 November 1993
#
#===========================================================================

if [ "$#" -eq 0 ]
then
    ps gv | egrep  "dcelocal|\
dfs|\
bos|\
fls|\
upserver|\
rpcd|\
cdsd|\
gdad|\
cdsadv|\
gdsd|\
secd|\
dtsd|\
sec_client" | \
    egrep -v "grep|SERVER"
else
    ps gv | egrep  "dcelocal|\
dfs|\
bos|\
fls|\
upserver|\
rpcd|\
cdsd|\
gdad|\
cdsadv|\
gdsd|\
secd|\
dtsd|\
sec_client" | \
    egrep -v "grep|SERVER|check_dce_memory" | \
    egrep $1

fi
```

Appendix E: Listing of Typical DCE Enviornment Status Report

```
DATE:    20    13    15    36
procs    memory            page              faults        cpu
----- ----------- ----------------------- ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0  7107   393   0   0   0   0    0   0 128  176  58  5  3 30  0
8    paging00
21   hd6
  7702      - S     0:09  106   808  1096 32768   377   196  0.6  2.0 cdsadv
  9885  pts/0 S     0:01    0   748   852 32768    30    40  0.1  1.0 sec_clientd
 11204  pts/0 S     1:00   16  1516  3632 32768  1465  2028  3.7  6.0 secd -bootstrap
 11437      - S     0:22   26   868  1064 32768   115   128  1.3  2.0 rpcd
 12327      - S     0:29   37  1408  2496 32768   745  1020  1.9  4.0 cdsd -a
 13345      - S     0:35   55  1240  1640 32768   363   324  2.2  3.0 /opt/dcelocal/bin/
cdsclerk -U root -u 0 -m 8198
 17826      - S     0:07   31   900  1220 32768   284   244  0.5  2.0 dtsd
 19017      - S     0:05    1   868  1384 32768   292   440  0.3  2.0 gdad
RPC DB files in /opt/dcelocal/var/rpc
total 56
-rw-r--r--   1 root     audit      24576 Nov 20 12:53 rpcdep.dat
-rw-r--r--   1 root     audit       4096 Nov 20 12:48 rpcdllb.dat
Security DB files in /opt/dcelocal/var/security/rgy_data
total 336
-rw-------   1 root     audit       2828 Nov 20 12:48 acct
-rw-------   1 root     audit      14960 Nov 20 12:48 acl
-rw-------   1 root     audit       3456 Nov 20 12:48 group
-rw-------   1 root     audit         32 Nov 20 12:48 master_info
-rw-------   1 root     audit        644 Nov 20 12:48 org
-rw-------   1 root     audit       5356 Nov 20 12:48 person
-rw-------   1 root     audit        324 Nov 20 12:48 replicas
-rw-------   1 root     audit       2204 Nov 20 12:48 rgy
-rw-------   1 root     audit         68 Nov 20 12:48 rgy_state
-rw-------   1 root     audit     115255 Nov 20 13:15 update_log
-rw-r--r--   1 root     audit         31 Nov 20 12:48 /krb5/krb.conf
-rw-------   1 root     audit        297 Nov 20 12:49 /krb5/v5srvtab
CDS DB files in /opt/dcelocal/var/directory/cds
total 256
drwxr-xr-x   4 root     bin          512 Nov 20 12:50 adm
-rw-------   1 root     audit         54 Nov 20 12:49 cds_files
-rw-------   1 root     audit       4608 Nov 20 12:49 nashua.cell#nashua_ch.check-
point0000000001
-rw-------   1 root    audit    105984 Nov 20 12:53 nashua.cell#nashua_ch.tlog0000000001
-rw-------   1 root     audit         11 Nov 20 12:49 nashua.cell#nashua_ch.version
-rw-r--r--   1 root     audit        300 Nov 20 12:49 server_mgmt_acl.dat
```

Appendix F: Partial Listing of an Example `testlog`

```
Script command is started on Wed Nov 10 15:49:50 CST 1993.tucson root#
tucson root#
tucson root# dce_login cell_admin -dce-
Password must be changed!
tucson root# /DCESuitcase/bin/idexpire
The Current Date/Time is: 15:53:37 on 11/10/93
Your Identity Expires: at 15:52:39 on 11/20/93
tucson root#  cd /tmp/STRESS/TESTS
tucson root#
tucson root# cds_test_driver

You must have already started the script program to capture the
output from this test suite.

 Press Return/Enter to continue, or Control-C to stop.



Checking to be sure you are dce_login'd as cell_admin.
Logging CDS response time every 60 seconds
to /tmp/cds.check.log ...


You will run cases #1 to #502 which will add a total of 50200 entries
to the CDS database.


Creating CDS directory /.:/hosts/test
cds_test_driver: Running CASE 1
DATE:    10    15    54    41
procs    memory             page              faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0  8639  1260   0   1   0   3   17   0 128 3991  54 33 35 29  4
10    paging00
25    hd6
  9131      - S     0:53    3  1688  2316 32768   588   548  1.1  4.0 cdsd -a
 11181      - S     0:57    1  1520  1880 32768   285   280  1.1  3.0 /opt/dcelocal/bin/
cdsclerk -U (0,10) -u 0 -m 12293
 12777      - S     0:11    1  1128  1384 32768   203   200  0.2  2.0 dtsd
 14573      - S     0:01    0   972  1076 32768    29    36  0.0  2.0 sec_clientd
 18544      - S     0:07    0  1056  1216 32768    88    92  0.1  2.0 rpcd
 20129      - S     0:06   76   932  1212 32768   289   196  0.1  2.0 cdsadv
 22175      - S     0:19    8  1340  2220 32768  1160   800  0.4  4.0 secd -bootstrap
RPC DB files in /opt/dcelocal/var/rpc
total 64
drwxr-xr-x   3 root     system       512 Nov  1 15:14 adm
-rw-r--r--   1 root     audit      24576 Nov 10 14:36 rpcdep.dat
-rw-r--r--   1 root     audit       4096 Nov 10 14:28 rpcdllb.dat
Security DB files in /opt/dcelocal/var/security/rgy_data
total 160
-rw-------   1 root     audit       2828 Nov 10 14:28 acct
-rw-------   1 root     audit      14960 Nov 10 14:28 acl
-rw-------   1 root     audit       3456 Nov 10 14:28 group
-rw-------   1 root     audit         32 Nov 10 14:28 master_info
-rw-------   1 root     audit        644 Nov 10 14:28 org
-rw-------   1 root     audit       5392 Nov 10 14:28 person
-rw-------   1 root     audit        324 Nov 10 14:28 replicas
```

```
-rw-------   1 root     audit       2204 Nov 10 14:28 rgy
-rw-------   1 root     audit         68 Nov 10 14:28 rgy_state
-rw-------   1 root     audit      28465 Nov 10 15:13 update_log
-rw-r--r--   1 root     audit         46 Nov 10 14:28 /krb5/krb.conf
-rw-------   1 root     audit        230 Nov 10 14:30 /krb5/v5srvtab
CDS DB files in /opt/dcelocal/var/directory/cds
total 288
drwxr-xr-x   3 root     audit        512 Nov  1 15:16 adm
-rw-------   1 root     audit         54 Nov 10 14:31 cds_files
-rw-r--r--   1 root     audit        315 Nov 10 14:31 server_mgmt_acl.dat
-rw-------   1 root     audit       4608 Nov 10 14:31 tucson.cell#tucson_ch.check-
point0000000001
-rw-------  1 root    audit    120320 Nov 10 15:54 tucson.cell#tucson_ch.tlog0000000001
-rw-------   1 root     audit         11 Nov 10 14:31 tucson.cell#tucson_ch.version
cds_test_case: Output file is /tmp/cds.test.21223
rpccp>
>>> binding information and objects exported

...
    { 98 messages omitted. }
...
rpccp>
>>> binding information and objects exported

rpccp> cds_test_driver: Running CASE 2
DATE:    10    15    58    20
procs    memory             page              faults        cpu
----- ----------- ----------------------- ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0  8742  1040   0   1   0   3   17   0 128 3991  54 33 35 29  4
11    paging00
25    hd6
  9131      - S     1:24    4  1888  2516 32768   588   548  1.6  4.0 cdsd -a
 11181      - S     1:24    1  1612  1972 32768   285   280  1.6  3.0 /opt/dcelocal/bin/
cdsclerk -U (0,10) -u 0 -m 12293
 12777      - S     0:12    1  1128  1384 32768   203   200  0.2  2.0 dtsd
 14573      - S     0:01    0   972  1076 32768    29    36  0.0  2.0 sec_clientd
 18544      - S     0:07    0  1056  1216 32768    88    92  0.1  2.0 rpcd
 20129      - S     0:06   76   932  1212 32768   289   196  0.1  2.0 cdsadv
 22175      - S     0:19    8  1340  2220 32768  1160   800  0.4  4.0 secd -bootstrap
RPC DB files in /opt/dcelocal/var/rpc
total 64
drwxr-xr-x   3 root     system       512 Nov  1 15:14 adm
-rw-r--r--   1 root     audit      24576 Nov 10 14:36 rpcdep.dat
-rw-r--r--   1 root     audit       4096 Nov 10 14:28 rpcdllb.dat
Security DB files in /opt/dcelocal/var/security/rgy_data
total 160
-rw-------   1 root     audit       2828 Nov 10 14:28 acct
-rw-------   1 root     audit      14960 Nov 10 14:28 acl
-rw-------   1 root     audit       3456 Nov 10 14:28 group
-rw-------   1 root     audit         32 Nov 10 14:28 master_info
-rw-------   1 root     audit        644 Nov 10 14:28 org
-rw-------   1 root     audit       5392 Nov 10 14:28 person
-rw-------   1 root     audit        324 Nov 10 14:28 replicas
-rw-------   1 root     audit       2204 Nov 10 14:28 rgy
-rw-------   1 root     audit         68 Nov 10 14:28 rgy_state
-rw-------   1 root     audit      28465 Nov 10 15:13 update_log
-rw-r--r--   1 root     audit         46 Nov 10 14:28 /krb5/krb.conf
-rw-------   1 root     audit        230 Nov 10 14:30 /krb5/v5srvtab
CDS DB files in /opt/dcelocal/var/directory/cds
total 1096
```

```
drwxr-xr-x   3 root     audit         512 Nov  1 15:16 adm
-rw-------   1 root     audit          54 Nov 10 14:31 cds_files
-rw-r--r--   1 root     audit         315 Nov 10 14:31 server_mgmt_acl.dat
-rw-------   1 root     audit        4608 Nov 10 14:31 tucson.cell#tucson_ch.check-
point0000000001
-rw-------  1 root    audit    536576 Nov 10 15:58 tucson.cell#tucson_ch.tlog0000000001
-rw-------   1 root     audit          11 Nov 10 14:31 tucson.cell#tucson_ch.version
cds_test_case: Output file is /tmp/cds.test.21001
rpccp>
>>> binding information and objects exported

...
```

Appendix G: Listing of the `CheckCDS` Script

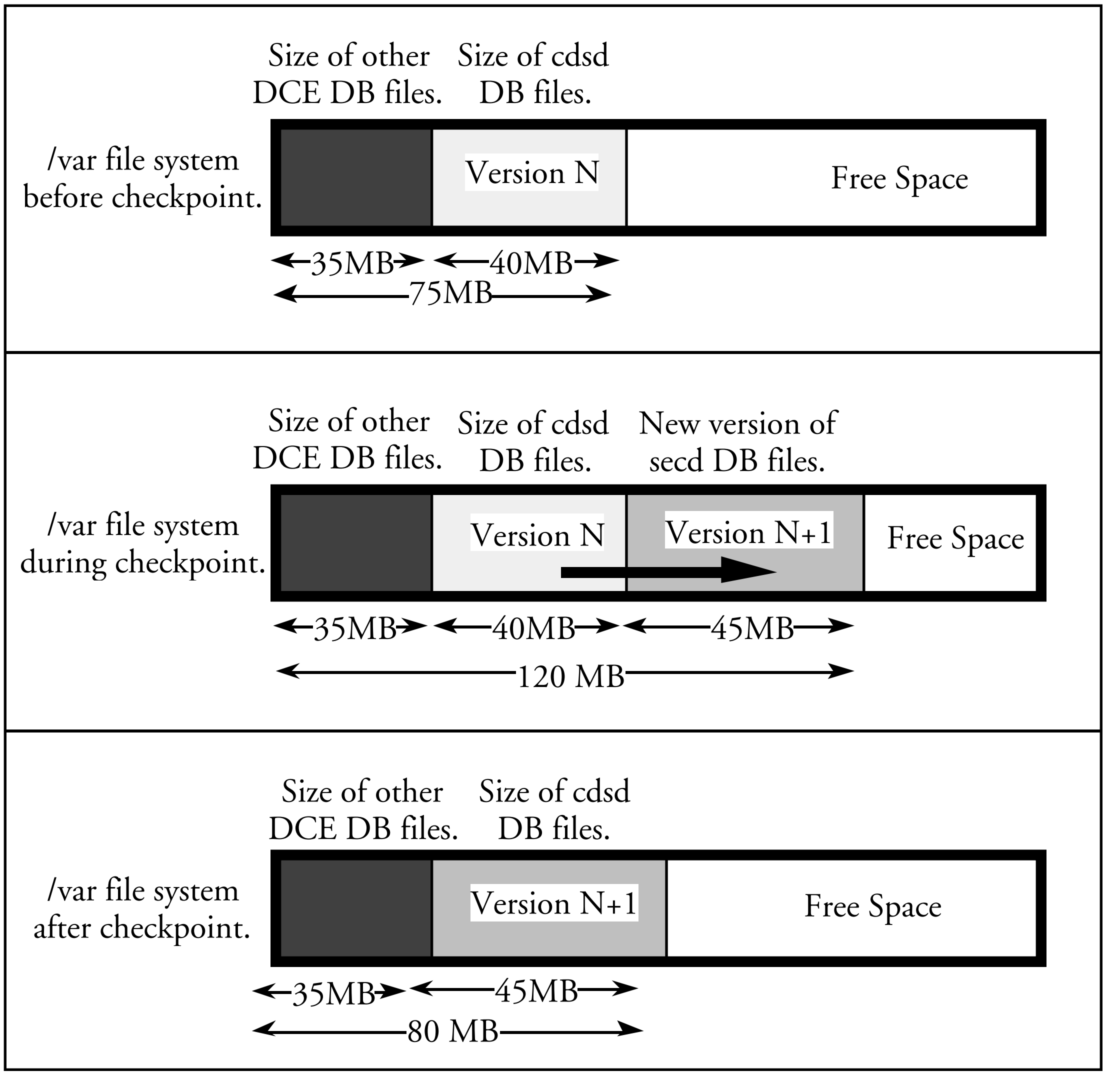
```
#!/bin/sh

CMD=`basename $0`
if [ $# -eq 3 ]
then
    FORCE=0
    shift
else
    FORCE=1
fi

if [ $# -ne 2 ]
then
        echo "

  Usage $CMD <seconds> <outputfile>
  Usage $CMD -force <seconds> <outputfile> &


  To log CDS response time every <seconds> to <outputfile>.
  Use the -force flag to run the program in the background.

"
        exit
else
        DURATION=$1
        OUTPUTFILE=$2
        if [ -f $OUTPUTFILE -a $FORCE -eq 1 ]
    then
        echo -n "Remove existing file \"$OUTPUTFILE\"? " > /dev/tty
        read a
        case $a in
            y|Y|Yes|yes)
                /bin/rm -f $OUTPUTFILE
            ;;
            *)
            ;;
        esac
    fi

fi

echo "Logging CDS response time every $DURATION seconds
to $OUTPUTFILE ..." > /dev/tty

CheckDCE()
{
        echo -n "DATE: " >> $OUTPUTFILE
        date "+%d %H %M %S" >> $OUTPUTFILE
        cdscp show cell /.: | egrep "Clearinghouse Name|Error" >> $OUTPUTFILE
}

while :
do
        StartClock=`date "+%d %H %M %S"`
        StartDate=`echo $StartClock | awk '{printf ("%d", $1)}'`
        StartHour=`echo $StartClock | awk '{printf ("%d", $2)}'`
        StartMin=`echo $StartClock | awk '{printf ("%d", $3)}'`
```

```
        StartSec=`echo $StartClock | awk '{printf ("%d", $4)}'`

        CheckDCE

        EndClock=`date "+%d %H %M %S"`
        EndDate=`echo $EndClock | awk '{printf ("%d", $1)}'`
        EndHour=`echo $EndClock | awk '{printf ("%d", $2)}'`
        EndMin=`echo $EndClock | awk '{printf ("%d", $3)}'`
        EndSec=`echo $EndClock | awk '{printf ("%d", $4)}'`

     ElapsedTime=`echo "$StartDate $EndDate $StartHour $EndHour $StartMin $EndMin $Start-
Sec $EndSec" | \
    awk '{printf("%d", (($2-$1)*216000) + (($4-$3)*3600) + (($6-$5)*60) + ($8-$7)  )}' `

        echo "ELAPSED time is: $ElapsedTime seconds." >> $OUTPUTFILE

        SleepTime=`echo $DURATION $ElapsedTime | awk '{printf ("%d", $1 - $2)}'`

        if [ $SleepTime -gt $DURATION ]
        then
                sleep $DURATION
        elif [ $SleepTime -lt 0 ]
        then
            echo "SLEEP ERROR: This log ended after next one should begin" >> $OUTPUTFILE
        elif [ $SleepTime -gt 0 ]
        then
                sleep $SleepTime
        fi

done

exit
```

Appendix H: Partial Lsting of and Example `checklog`

```
•••
ELAPSED time is: 2 seconds.
DATE: 10 18 23 02
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 3 seconds.
DATE: 10 18 24 02
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 2 seconds.
DATE: 10 18 25 04
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 3 seconds.
DATE: 10 18 26 04
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 2 seconds.
DATE: 10 18 27 06
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 4 seconds.
DATE: 10 18 28 06
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 2 seconds.
DATE: 10 18 29 07
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 3 seconds.
DATE: 10 18 30 08
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 3 seconds.
DATE: 10 18 31 09
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 1 seconds.
DATE: 10 18 32 11
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 2 seconds.
DATE: 10 18 33 12
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 95 seconds.
SLEEP ERROR: This log ended after next one should begin
DATE: 10 18 34 48
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 3 seconds.
DATE: 10 18 35 48
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 3 seconds.
DATE: 10 18 36 49
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 2 seconds.
DATE: 10 18 37 50
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 2 seconds.
DATE: 10 18 38 51
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 3 seconds.
DATE: 10 18 39 52
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 2 seconds.
DATE: 10 18 40 54
          Clearinghouse Name = /.../tucson.cell/tucson_ch
ELAPSED time is: 2 seconds.
•••
```

Appendix I: Why Checkpoints Cause Large, Transient File Size Increases



CDS Checkpoints cause a large but transient increase in file system usage. Picture a situation in which the size of all the DCE database files, excluding the CDS checkpoint file, totals 35 MB, and in which the checkpoint file is 40MB just prior to a checkpoint that will add 5MB of data. The checkpoint will result in a temporary surge of file system usage of 35MB + 40MB + 45MB = 120MB. After the new CDS checkpoint file is successfully written, the old version of the file is removed. Although the final usage is only 5MB more than the initial usage, the DCE Administrator must allow “head room” in the `/var` file system. Certain DCE daemons die when and if the `/var` file system fills during a checkpoint.