CITI Technical Report 94–1

Adding 50,200 Users to a DCE Registry: 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 "human users" to the cell registry. 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 `secd` paging space, registry file space usage, and `rgy_edit` response time. Although resource consumption increases linearly with the number of users added to the cell registry, `rgy_edit` response time increases markedly during a registry checkpoint operation. The registry performance statistics included in this study enable the DCE administrator to establish a resource usage baseline. However, further investigation is required to ascertain whether security server replication will smooth the `rgy_edit` response time or exacerbate `rgy_edit` delays.

February 14, 1994

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

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

Mark R. Carter

February 14, 1994

1. Introduction

At the Center for Information Technology Integration (CITI), we hypothesize that a large increase in the number of users in a Distributed Computing Environment (DCE) cell will result in a linear increase in paging space usage and registry database file sizes. We further hypothesize that registry access times via `rgy_edit add` and `rgy_edit view` commands will remain fairly constant as the registry 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 registry, are also addressed:

- *Feasibility*—Can an administrator add 50,200 users (principals and accounts) to the registry?
- *Time-Based Issues*—How long does it take to add 50,200 users? Does it take longer to add an individual user as the size of the cell registry increases? Does it take longer to look up a user as the size of the cell registry increases?
- *Paging Space Issues*—How much does paging space usage increase when adding 50,200 users? What is the paging space usage per individual user 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 users? What is the file storage size per individual user during the add sequence? Does the file storage size change after the add sequence is completed?

The registry service is not placed under any load except that required to add the users to the database. The `rgy_edit add` commands are used to add users to the registry, and the `rgy_edit view` commands are used to periodically check `rgy_edit` response time, but `dce_logins` are performed during the registry 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:

- 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.
- The "Maximum possible UID" was changed to 55,000. Without this modification, the maximum number of users that can be added to the cell is approximately 32,664.

2.3 The Why and How of 50,200 users

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 principal names.

To create a file with enough unique identifiers (organizations, groups, principals, accounts) 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 identifiers, 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 users. (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 users are added to the registry in alphabetical order. This is by design to ascertain whether this diabolical insertion order causes any problems. It does not.

2.4 A User

Adding a user to the registry means adding a DCE principal and account in the registry.

2.5 The Registry Test Case

Our registry test case adds one new organization, one new group, 100 principals, and 100 accounts to the registry. Every group of 100 accounts is created with the 100 corresponding server principal names obtained from the input file, and a group and organization name derived from the first name in the file. Creating a new group and organization for each batch of 100 accounts more closely simulates actual conditions than does creating all 50,200 accounts with the same group and organization.

The test case accepts a list of words as input. Because the `rgy_edit` command line interface is slow, each test case constructs a temporary file that contains `rgy_edit` commands in the form of a `here` document. The file of `rgy_edit` 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 Registry Test Suite

There are 502 test cases in the test suite. The registry test suite adds 50,200 users to the registry.

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 users from the time stamp in the status report, and to determine the number of users in the registry when the status is obtained.

While the test suite is running, another process monitors `rgy_edit` response time with a `rgy_edit -a -v cell_admin` command once per minute.

The *check time* is the time required to execute one `rgy_edit -a -v cell_admin` command. Check times are reported in seconds.

3. Procedure

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

3.1 Registry Test Suite Execution

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

Please see Appendix A for a listing of the `rgy_test_driver` script and Appendix B for a listing of the `rgy_test_case` script. 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 `rgy_edit` program prints, are collected in the `testlog` file using the `script` command.

The size of the `testlog` files is typically about 1.3 MB and they contain output captured from the screen during test suite execution. This output consists of a mixture of the status report and `rgy_edit` feedback. See Appendix F for a partial listing of an example `testlog`.

3.3 Monitoring the Test Cases—CheckRGY

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

At regular intervals, the `CheckRGY` script issues a `rgy_edit -v -a cell_admin` command. The output from the `rgy_edit` command, and the execution time are recorded in the `checklog` file by the `CheckRGY` script. If the `rgy_edit` 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 `rgy_edit` command execution time. This results in a more regular sample interval than does a simple `sleep N` because the duration of the `rgy_edit` command itself is considered.

The size of a `checklog` file is typically about 175 KB. See Appendix H for a partial listing of an example `checklog`.

4. Results

The registry test suite did not successfully complete all test cases.

Registry test cases began to fail on account number 85 of case number 473 (OSF), account number 90 of case number case 474 (IBM). The failures began with the error message: “Operation involving multiple registry updates was only partially successful,” followed by multiple error messages stating: “Registry service unavailable.” The error messages indicate and the status report confirms that `secd` crashed. In both cases, DCE daemons were halted with `/etc/dce.-clean` and restarted with `/etc/rc.dce`. Execution of the test suite was resumed with the test case in which the failure occurred. The test suite completed successfully without further incident.

After the test suite execution, we queried the registry to get a count of organizations, groups, principals, and accounts. The count totals were as expected. In other words, all organizations, groups, principals, and accounts that the test suite adds to the registry actually appear in the registry. Table 1 shows the number of items in the registry domain at the conclusion of the test suite. Domain entries added during cell configuration are not included in this count.

TABLE 1. Items in the Registry After the Test

Number of items in each registry domain	OSF	IBM
Organization (+502)	502	502
Groups (+502)	502	502
Principals (+50,200)	50,200	50,200
Accounts (+50,200)	50,200	50,200

The registry test suites may have failed to execute to completion because:

1. The `/opt/dcelocal/var` file system filled,
2. The machine’s paging space usage climbed too high, or
3. The virtual memory size of the `secd` process was too large.

Exploring these possibilities, we see that the `/opt/dcelocal/var` file system did not fill during test suite execution. The size of this file system was much greater than necessary. No `/var` file system reached more than 50% full during test suite execution. Moreover, if this file system had filled, `secd` would not have successfully restarted. It would have resumed where it left off, found it could not write any data, and would have died again.

Additionally, the paging space usage did not climb too high. The registry tests failed when paging space usage was approximately 160 MB, 70% of available paging space. At first glance, this suggests that `secd` experienced a catastrophic paging fault and died. However, after the paging space was increased from 228 MB to 308 MB (IBM), the test suite failed again. The first time, the test suite failed on account number 47,485, and the second time, the test suite failed on account number 47,477. Increasing paging space did not prevent `secd` from crashing at almost the same place (IBM). (We believe we would observe the same result on the OSF release.)

However, it appears that `secd` doesn’t require all the memory it obtains. When `secd` crashed, the virtual size of the data section (`SIZE` column of `ps gv`) of the `secd` process was 129.4 MB

(OSF) and 129.8 MB (IBM). AIX imposes a 128 MB limit on the data section of a process, and the `SIZE` of the `secd` processes exceeded this limit. Another finding suggests an underlying problem with the `secd` code. When `secd` was restarted after it crashed, paging space consumption and virtual memory usage were cut in half, yet no registry data were lost.

4.1 Time to Execute the Complete Test Suite

The execution time of the registry test suite totaled 21.4 hours for OSF and 18.8 hours for IBM.

The overall average time for adding one user to the registry was 1.53 seconds for OSF and 1.35 seconds for IBM. The graph in Figure A shows the serial number of the test case (OSF and IBM) across the elapsed time.

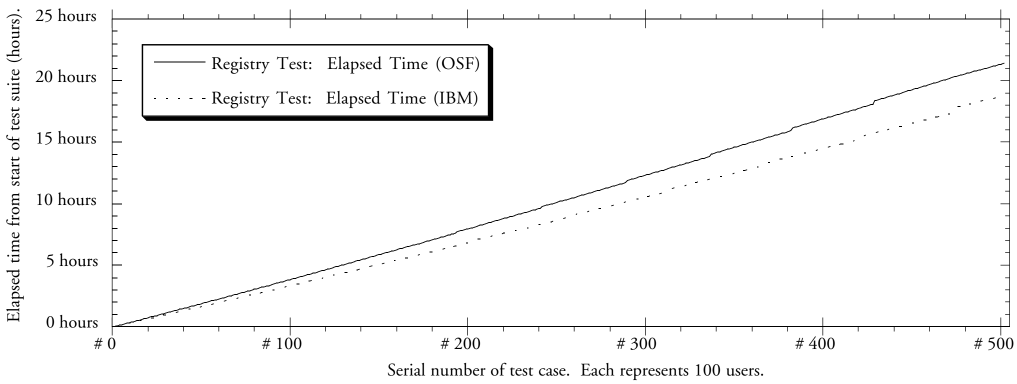

Figure A. Test Suite Execution Time

4.2 Case Times

The overall average case time was 154 seconds for OSF and 135 seconds for IBM.

On average, test cases execute slightly faster on the IBM release. Case times increased only slightly as the size of the registry increased (Table 2). The last sample was obtained after `secd` was restarted. The time required to add users decreased following the restart.

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	137	120
51-100	141	122
101-150	145	126
151-200	151	126
201-250	154	131
251-300	160	136
301-350	163	141
351-400	166	143
401-450	171	148
451-500	149	156

However, the maximum case time was 960 seconds for OSF and 1,389 seconds for IBM. The longest observed case time occurred on the IBM release, and required over 23 minutes to complete. Figure B shows registry test case times (OSF and IBM) graphed against the elapsed time from the beginning of test suite.

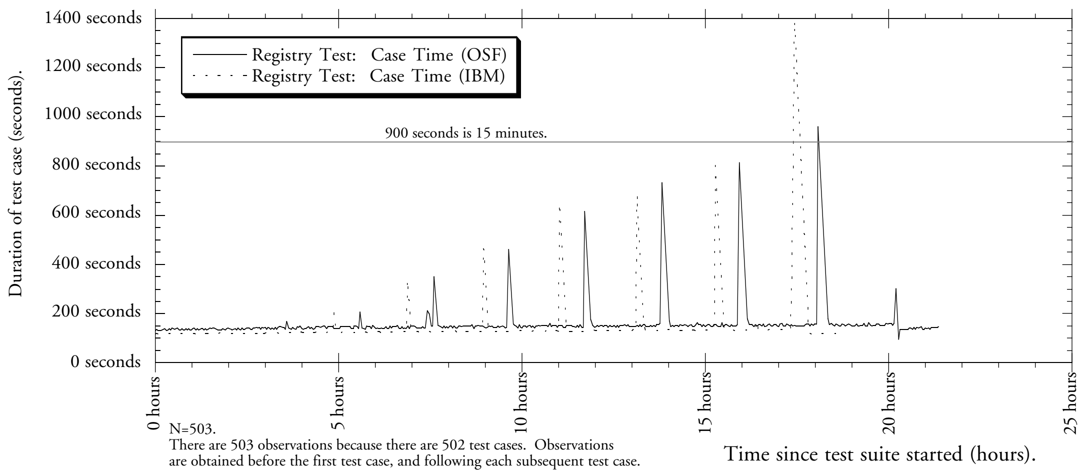

Figure B. Case Times

Occasional lengthy case times (> 400 seconds) occur in both releases. The lengthy times cluster around registry checkpoints—when `secd` 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	147	127
including slowdowns	154	135

TABLE 4. Maximum Case Times

	Maximum OSF case time in seconds	Maximum IBM case time in seconds
excluding slowdowns	172	152
including slowdowns	960	1,389

Tables 5 and 6 show frequency distributions of case times during testing on the OSF release. Of the 502 total observations, 492 test cases (98%) were completed in less than 201 seconds and only 9 (3%) took longer than 202 seconds to complete. Moreover, 478 test cases (95%) completed between 130 and 160 seconds.

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

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	1	101	1	.2%	
2	101	201	491	98%	- Mode
3	201	301	2	.4%	
4	301	401	2	.4%	
5	401	501	1	.2%	
6	501	601	0	0%	
7	601	701	1	.2%	
8	701	801	1	.2%	
9	801	901	1	.2%	
10	901	1001	1	.2%	

Table 6. Frequency Distribution of the Time (in seconds) Required to Complete Cases Between 100 and 200 seconds (OSF)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	100	110	0	0%	
2	110	120	0	0%	
3	120	130	0	0%	
4	130	140	92	18.4%	
5	140	150	225	44.9%	- Mode
6	150	160	161	32.1%	
7	160	170	8	1.6%	
8	170	180	3	.6%	
9	180	190	1	.2%	
10	190	200	1	.2%	

Tables 7 and 8 show frequency distributions of case times during testing on the IBM release. Of the 502 total observations, 493 test cases (98.4%) were completed in less than 201 seconds and only 8 (3%) took longer than 202 seconds to complete. Moreover, 485 test cases (97%) were completed between 110 and 140 seconds.

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

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	1	101	0	0%	
2	101	201	493	98.4%	- Mode
3	201	301	2	.4%	
4	301	401	1	.2%	
5	401	501	1	.2%	
6	501	601	0	0%	
7	601	701	2	.4%	
8	701	801	0	0%	
9	801	901	1	.2%	
10	901	1001	0	0%	

Table 8. Frequency Distribution of the Time (in seconds) Required to Complete Cases Between 100 and 200 seconds (IBM)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	100	110	0	0%	
2	110	120	61	12.2%	
3	120	130	286	57.1%	- Mode
4	130	140	139	27.7%	
5	140	150	5	1%	
6	150	160	1	.2%	
7	160	170	0	0%	
8	170	180	1	.2%	
9	180	190	0	0%	
10	190	200	0	0%	

The checkpoint operation takes longer as the size of the registry increases. Therefore, the larger the size of the registry, 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 an individual `rgy_edit add` command does not block because of the checkpoint. Instead, the duration of all `rgy_edit` commands increases.

4.3 Paging Space Usage

Paging space consumption increases linearly with the number of users added to the registry, but decreases considerably when DCE services are restarted at the end of the test suite or after a `secd` *crash.*

Paging space usage increases 128 MB for OSF and 126 MB for IBM. Before `secd` crashed, the paging space growth was 2,708 bytes per user (OSF) and 2,645 bytes per user (IBM). After `secd` was restarted, the paging space growth during the remainder of the test suite was 1,508 bytes per user (OSF) and 1,365 bytes per user (IBM). Paging space usage was similar in both test conditions (OSF and IBM). Also in both test conditions, the minimum paging space usage occurs at the start of the test suite, and the maximum usage occurs not at the end of the test suite, but immediately before `secd` crashed.

Paging space usage approached approximately 70% of available space immediately prior to the crash. Tables 9 and 10 show a summary of paging space usage before and after `secd` crashed.

TABLE 9. Paging Space Usage Summary per User—Before `secd` Crash

	OSF	IBM
Number of Users added to registry (count)	47,385	47,490
Paging space usage at start of add sequence (MB)	28.64	34.24
Paging space usage at end (crash) of add sequence (MB)	156.96	159.92
Growth in paging space used (MB)	128.32	125.68
Growth rate (Bytes/user)	2,708	2,646

TABLE 10. Paging Space Usage Summary per User—After `secd` Crash

	OSF	IBM
Number of Users added to registry (count)	4,615	4,510
Paging space usage at start of add sequence (MB)	83.96	90.36
Paging space usage at end of add sequence (MB)	90.92	96.52
Growth in paging space used (MB)	6.96	6.16
Growth rate (Bytes/user)	1,508	1,365

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

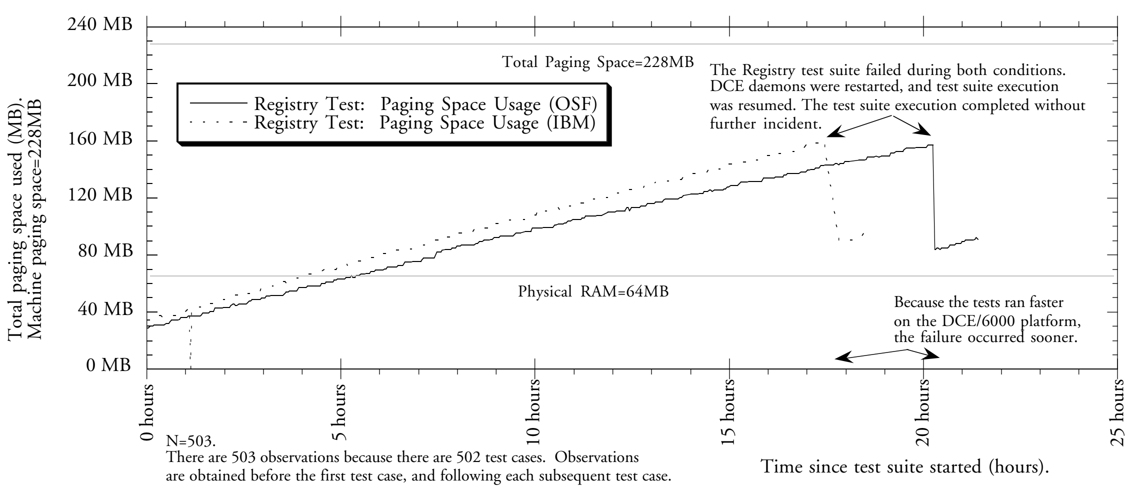

Figure C. Paging Space

Tables 11 and 12 show a summary of paging space usage per hour before and after `secd` crashed.

TABLE 11. Paging Space Usage Summary, per Hour, Before **`secd`** Crashed

	OSF	IBM
Elapsed time until `secd` crash (hours)	20.24	17.43
Paging space usage at start of add sequence (MB)	28.64	34.24
Paging space usage at end (crash) of add sequence (MB)	156.96	159.92
Growth in paging space used (MB)	128.32	125.68
Growth rate (MB/hour)	6.34	7.21

TABLE 12. Paging Space Usage Summary, per Hour, After Restarting **`secd`**

	OSF	IBM
Elapsed time after `secd` crash (hours)	1.12	0.90
Paging space usage at restart of add sequence (MB)	83.96	90.36
Paging space usage at end of add sequence (MB)	90.92	96.52
Growth after crash (MB)	6.96	6.16
Growth rate (MB/hour)	6.21	6.88

Paging space usage increased at almost the same rate per hour before `secd` crashed as after it was restarted. In contrast to paging space growth per hour, paging space usage per user decreases considerably after restarting `secd`.

When `secd` crashed and was subsequently restarted, paging space usage on the machine—measured at the end of the test case preceding the crash—decreased from a high of 157 MB (OSF), and 160 MB (IBM) to 84 MB (OSF) and 90 MB (IBM). When `secd` was restarted, the amount of paging space usage dropped by almost one half.

4.4 Paging Space After the Test Suite

After all entries were added to the registry, DCE core processes were halted and restarted. The paging space usage was measured immediately after DCE was restarted. Table 13 shows a summary of this paging space usage.

TABLE 13. 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)	28.64	31.72
Size of paging space after DCE restart (MB)	84.76	90.16
Difference from starting size.	56.12	58.44
Paging space usage after restart (Bytes/user)	1118	1164

As entries are added to the namespace, the average amount of paging space per user is 2,708 bytes for OSF and 2,646 bytes for IBM. However, after all entries are added and DCE is restarted, the average amount of paging space per user drops to 1,118 bytes (OSF) and 1,164 bytes (IBM).

4.5 Size of DCE Database Files

The combined size of the registry database files increases linearly with the number of users added to the registry. The size of other DCE database files remains almost constant as users are added to the registry.

The size of the registry database files increased 46 MB for OSF and IBM. The registry database files are `acct`, `acl`, `group`, `master_info`, `org`, `person`, `replicas`, `rgy`, `rgy_state`, and `update_log`, and they all reside in `/opt/dcelocal/var/security/rgy_data`. The size of the registry database file increased 922 bytes per user (OSF), and 924 bytes per user (IBM). The increase in size of the registry files is virtually equal on the two releases.

Tables 14 and 15 show the minimum and maximum combined size of all DCE database files and registry database files respectively.

TABLE 14. DCE Database File Size Summary

DCE database file size summary	OSF	IBM
Size of DB files at start of test suite (bytes)	201,074	204,919
Size of DB files at end of test suite (bytes)	49,012,104	48,966,341
Growth of DB files (bytes)	48,811,030	48,761,422
Size of DB files per user (bytes/user)	972.33	971.34

TABLE 15. Registry Database File Size Summary

DCE Security DB file size summary:	OSF	IBM
Size of security DB at start of test suite (bytes)	30,200	30,171
Size of security DB at end of test suite (bytes)	46,293,400	46,392,039
Growth of security DB (bytes)	46,263,200	46,361,868
Size of security DB per user (bytes/user)	921.58	923.54

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).

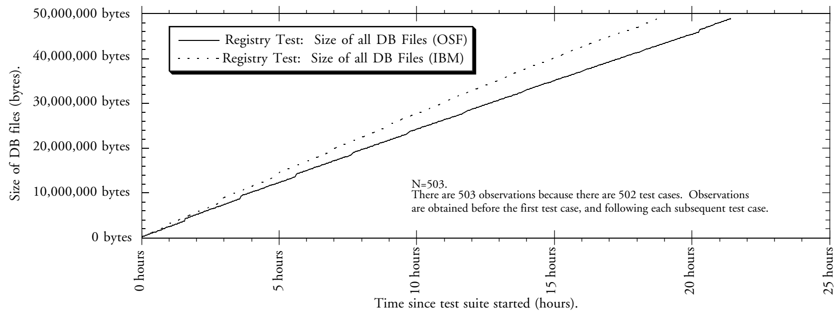

Figure D. DCE Database File Sizes

The graph in Figure E shows the combined registry database files across the elapsed time from the beginning of the test suite (OSF and IBM).

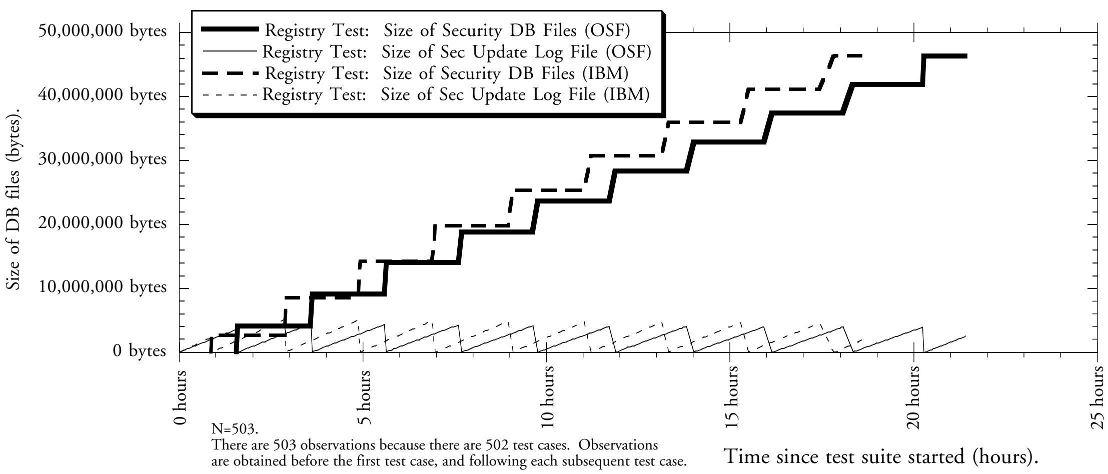

Figure E. Registry Database File Sizes

In both test conditions (OSF and IBM), the minimum size of the registry database files occurs at the beginning of the test suite, and the maximum size of the database 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 registry test suite execution, `CheckRGY` initiates a `rgy_edit -a -v cell_admin` command approximately once every minute. The time required to execute this command is called the check time. The time required to view a user remains virtually constant as the size of the registry increases—unless the registry 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 5.6 seconds, the median check time is 2.0 seconds, suggesting there are occasional lengthy check times. If the registry is engaged in a checkpoint operation, the check time increases considerably. Table 16 shows summary statistics on the check times. Tables 17 and 18 show a frequency distribution of the check times.

TABLE 16. Summary Statistics of the Check Time in Seconds (IBM)

Check time	Registry
Count (N=)	1086
Minimum time (seconds)	1
Maximum time (seconds)	194
Average time (seconds)	5.6
Median time (seconds)	2

Of the 1,086 total commands observed, 1,053 (97%) returned in less than 9 seconds, and only 33 (3%) took longer than 9 seconds. Moreover, 1,018 commands (93%) returned in 5 seconds or less, and 92.5 (85%) returned in 3 seconds or less.

Table 17. Frequency Distribution of the Check Time in Seconds (IBM)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	0	12.9	1053	97%	- Mode
2	12.9	25.9	2	.2%	
3	25.9	38.8	2	.2%	
4	38.8	51.7	0	0%	
5	51.7	64.7	2	.2%	
6	64.7	77.6	3	.3%	
7	77.6	90.5	0	0%	
8	90.5	103.5	2	.2%	
9	103.5	116.4	10	.9%	
10	116.4	129.3	4	.4%	
11	129.3	142.3	2	.2%	
12	142.3	155.2	1	.1%	
13	155.2	168.1	4	.4%	
14	168.1	181.1	0	0%	
15	181.1	194	1	.1%	

Table 18. Frequency Distribution of Check Times Less Than 13 Seconds (IBM)

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	1	3	925	85.2%	- Mode
2	3	5	88	8.1%	
3	5	7	5	.5%	
4	7	9	1	.1%	
5	9	11	0	0%	
6	11	13	0	0%	

Although check times are usually less than 95 seconds, some are significantly longer—up to 194 seconds. Figure F graphs the registry test check times against the 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).

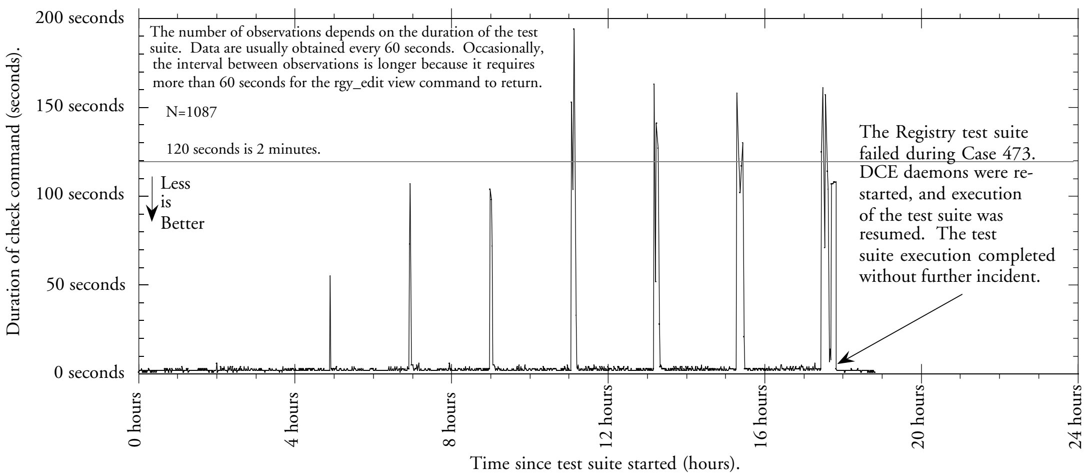

Figure F. Check Times (IBM)

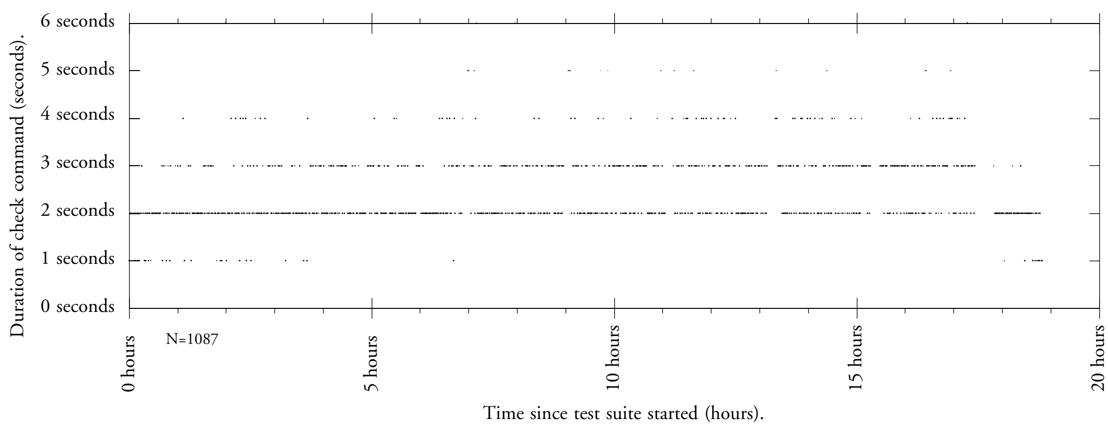

Figure G. Check Times Under 20 Seconds

Longer check times cluster around checkpoints, and occur more frequently as the registry size increases. Although the majority of check times are less than 6 seconds, Figure G reveals a trend toward slightly longer times. Data from the last hour were obtained after `secd` was restarted, and the check time decreased following the restart.

4.7 Why the Delays?

Case times and check time are substantially greater when the registry 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 `rgy_edit` response time and the size of the registry update log, even though the absolute magnitude of the values of each statistic ranges over many orders of magnitude.

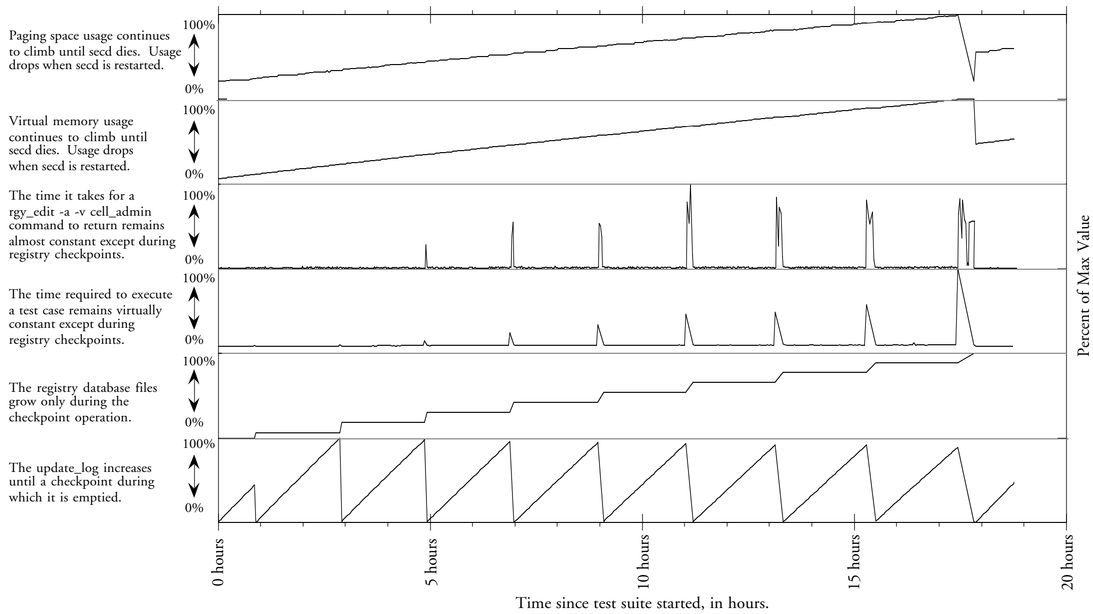

Figure H. Maximum Values (IBM)

This graph contains six layers, one upon the next. Layer 1 (saw tooth) is on the bottom, and represents the size of the registry update log file. Layer 2 (stair step) represents the combined size of all the security database files except the update log. When the registry file size increases, the update log size decreases. This is evidence of a registry checkpoint.

Layer 3 represents case times. Layer 4 represents check times. Longer than usual times are demonstrated by the spikes that occur while the registry 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. Notice that the usage drops when `secd` crashes and is restarted.

4.8 Checkpoint Frequency During the Test

Registry checkpoints are frequent when the registry is heavily exercised.

The registry test suite adds 50,200 users to the registry as quickly as possible. During testing, the registry checkpointed 10 times (OSF) and 9 times (IBM). On average, checkpoints occurred every 2.07 hours (OSF) and 2.11 hours (IBM). The graph in Figure I, shows registry intercheckpoint times graphed against the checkpoint number (OSF and IBM).

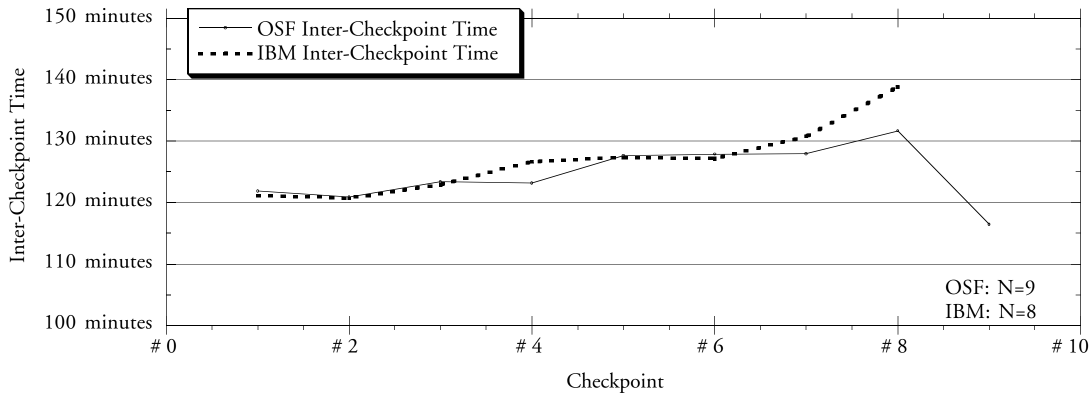

Figure I. Registry Intercheckpoint Times

On average, checkpoints occurred after the addition of 4,789 users (OSF) and 5,588 users (IBM). Figure J shows the registry intercheckpoint entry counts graphed against the checkpoint number. In this case, the entries are users and are rounded to the nearest 100.

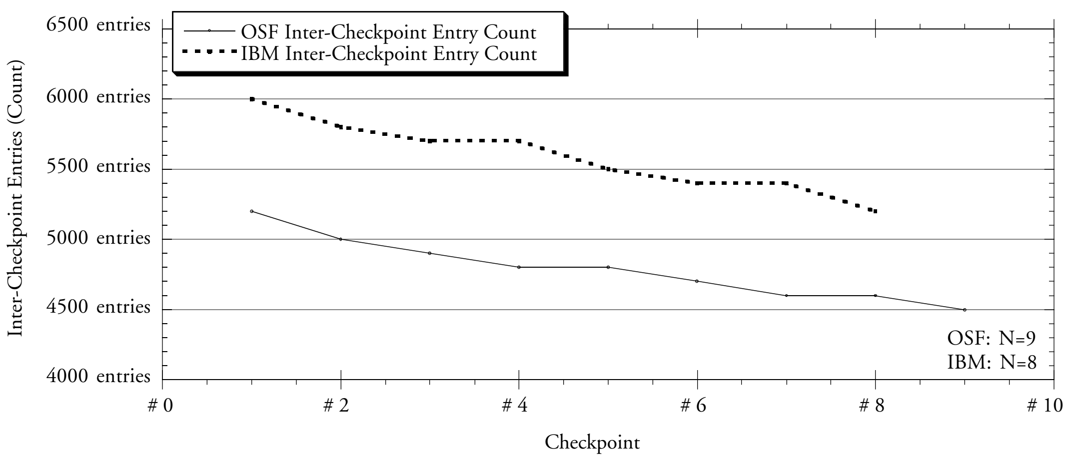

Figure J. Registry Intercheckpoint Entry Counts

Intercheckpoint times are similar on the OSF and IBM releases, and tend to increase as the size of the registry increases. This may happen because `secd` does not include the time required for the checkpoint operation in the calculation of when to checkpoint.

Intercheckpoint entry (user) counts are only slightly different on the two releases. More users are added between checkpoints on the IBM release than on the OSF release. The count tends to decrease as the size of the namespace increases. This suggests that checkpoints are based on time rather than the number or size of users added.

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 registry was not changed. Table 19 shows a frequency distribution of the check times (in seconds) in an idle cell.

Table 19. Frequency Analysis of Check Times in an "Idle" Cell Over a Six Day Period

Bar:	From: (>)	To: (≤)	Count:	Percent:	
1	0	4	3946	94.2%	- Mode
2	4	8	173	4.1%	
3	8	12	21	.5%	
4	12	16	7	.2%	
5	16	20	10	.2%	
6	20	24	10	.2%	
7	24	28	7	.2%	
8	28	32	6	.1%	
9	32	36	3	.1%	
10	36	40	4	.1%	
11	40	44	3	.1%	

Of the 4,190 observations, most (98%) are 8 seconds or less, and many (94.2%) are 4 seconds or less. No check times are longer than 44 seconds. The registry checkpointed occasionally during this observation—but the dates and sizes of the registry database files remained unchanged.

5. Conclusion

It requires 21.4 hours for OSF and 18.8 hours for IBM, to add 50,200 users to a DCE cell registry.

Registry size does not affect case times (the time required to add 100 users to the registry) unless the registry is checkpointing. Excluding checkpoints, the average case time is 147 seconds (OSF) and 127 seconds (IBM). The maximum case time is 172 seconds (OSF) and 152 seconds (IBM). Registry checkpoints diminish `rgy_edit add` performance significantly. Including checkpoints, the average case time is 154 seconds (OSF) and 135 seconds (IBM). The maximum case time increases to 960 seconds (OSF) and 1389 seconds (IBM).

Registry size does not affect the check times (the time required for a `rgy_edit -a -v cell_admin` command to return) unless the registry is checkpointing. The median check time is 2.0 seconds. Registry checkpoints increase check times significantly. The average check time is 5.6 seconds, and the maximum check time is 194 seconds. Check times were obtained only on the IBM release. We believe we would observe similar results on the OSF release.

Registry checkpoints (when `secd` saves its entire database from virtual memory to disk) are frequent when the registry is under heavy load. On average, checkpoints occur every 2.07 hours (OSF) and 2.11 hours (IBM). However, registry checkpoints are infrequent when the cell is not in use. When no changes were made to a cell registry for six days, few registry 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 users are added to the registry. The paging space usage increased 128 MB (OSF) and 126 MB (IBM). The paging space usage per individual user reached 2,708 bytes per user (OSF) and 2,646 bytes per user (IBM), during the add sequence. However, `secd` crashed when the virtual size of the data section (`SIZE` column of `ps gv`) of the `secd` process reached 129.4 MB (OSF) and 129.8 (IBM). AIX imposes a 128 MB limit on the data section of a process, and the `SIZE` of the `secd` processes exceeded this limit.

After `secd` was restarted, the paging space usage dropped to 1,508 bytes per user (OSF) and 1,365 bytes per user (IBM), during the remainder of the test suite. The paging space usage dropped to a low of 1,118 bytes per user (OSF) and 1,164 bytes per user (IBM), when the DCE daemons were restarted after the add sequence was completed.

Before `secd` crashed, the registry database files grew approximately 46 MB (OSF and IBM), yet the `SIZE` of the `secd` process and the paging space usage increased approximately 128 MB (OSF and IBM). When `secd` was restarted after it crashed, paging space consumption and virtual memory usage was cut by one half, yet no registry data were lost. This suggests an underlying problem with the `secd` code: Why does usage drop when the daemons are restarted?

The size of most DCE database files remains virtually constant as the size of the registry increased. The size of the registry database files increased linearly as the size of the registry increased. The size of the update log increased until a checkpoint, and then returned to a baseline size. The size of all database files increased 49 MB (OSF and 1IBM). The registry database files account for most of the increase. The file space usage per individual user during the add sequence is 972 bytes (OSF) and 971bytes (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 users to a single DCE cell, one can expect linear resource consumption as a result. Because the security server maintains a working copy of the registry database in virtual memory, `secd` may crash and have to be restarted. The need to maintain the entire registry in core memory questions the viability of the registry service in a large-scale cell. Delays associated with registry checkpoints increase response time while the users are being added, and periodically thereafter. Further investigation is required to ascertain whether security server replication will smooth the `rgy_edit` response time or exacerbate `rgy_edit` delays. That a read-only copy of the registry data is made available by replication suggests improvement in `rgy_edit` lookup time during checkpointing. However, `rgy_edit adds` may remain slow because they are addressed to the master (writable) replica. Moreover, the increased overhead associated with registry replication may offset any possible improvement.

Acknowledgments

This work was supported by IBM.

AppendixA: `rgy_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
}

rgy_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 Registry database.

"
while [ $CASE_NUMBER -le $MAX_NUMBER_OF_CASES ]
do
        # Set the name of the file from which to obtain the 100 names.
        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 "rgy_test_driver: Running CASE $CASE_NUMBER"

        # Capture the state of the DCE environment.
        write_log

        # Execute the test cases (add 100 principals and accounts).
        cat $FILE | rgy_test_case
        #cat $FILE | rgy_test_case2

        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
}

#==================================================================
# Main

prelim_check; # Be sure things are set for testing.

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

rgy_test_suite; # Run the test suite.

# Kill the check process to be sure it doesn't run all night!
sleep 100
kill -9 `ps xv | grep CheckRGY | grep -v grep | awk '{print $1}'`

exit 0
```

Appendix B: `rgy_test_case` Script

```
#!/bin/ksh

PW='-dce-'

TEMPFILE=/tmp/rgy.test.$$

echo "rgy_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
rgy_edit <<EOF"

GROUP="g_"${Names[0]}
ORG="o_"${Names[0]}

echo "do o
add $ORG
do g
add $GROUP"

echo "do p"
(( index=-1 ))
while (( index < $NUMBER ))
do
        (( index=$index+1 ))
        echo "add ${Names[$index]}"
done

echo "do a"
(( index=-1 ))
while (( index < $NUMBER ))
do
        (( index=$index+1 ))
    echo "add ${Names[$index]} -g $GROUP -o $ORG -pw $PW -mp $PW"
done

echo "quit"
echo "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
rgy_edit <<EOF
do o
add o_worrisome
do g
add g_worrisome
do p
add worrisome
add worry
add worse
add worsen
add worship
add worshipful
add worst
     {85 lines removed}
add yell
add yellow
add yellowish
add yelp
add yen
add yeoman
add yeomanry
add yeshiva
do a
add worrisome -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add worry -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add worse -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add worsen -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add worship -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add worshipful -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add worst -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
     {85 lines removed}
add yell -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add yellow -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add yellowish -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add yelp -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add yen -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add yeoman -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add yeomanry -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
add yeshiva -g g_worrisome -o o_worrisome -pw -dce- -mp -dce-
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 Sat Nov 20 13:08:11 EST 1993.nashua_root#
nashua_root#
nashua_root#
nashua_root# df
Filesystem    Total KB    free %used   iused %iused Mounted on
/dev/hd4         16384   10716   34%     779    19% /
/dev/hd2        135168     824   99%    9000    25% /usr
/dev/hd9var       4096    2504   38%      76     7% /var
/dev/hd3         24576   21148   13%     547     8% /tmp
/dev/hd1          4096    3816    6%      84     8% /home
/dev/lv00        53248   12320   76%    4018    28% /usr/vice/cache
/dev/lv01       253952  204460   19%     356     0% /opt
AFS           7200000072000000    0%       0     0% /afs
nashua_root# lsps -a
Page Space  Physical Volume   Volume Group    Size   %Used  Active  Auto  Type
paging00    hdisk0            rootvg         148MB       8     yes   yes    lv
hd6         hdisk0            rootvg          80MB      20     yes   yes    lv
nashua_root#
nashua_root# dl
nashua_root# dw
You are dce_login'ed as: cell_admin
nashua_root# exit
nashua_root# nashua_root#
nashua_root# dw
You are dce_login'ed as: hosts/nashua/self
nashua_root# dl
nashua_root# dw
You are dce_login'ed as: cell_admin
nashua_root# /DCESuitcase/bin/idexire
/DCESuitcase/bin/idexire: Command not found.
nashua_root# /DCESuticase/bin/idexpire
/DCESuticase/bin/idexpire: Command not found.
nashua_root# idexpire
The Current Date/Time is: 18:11:35 on 11/20/93
Your Identity Expires: at 13:10:17 on 11/30/93
nashua_root#
nashua_root# rgy_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.
Can't find CheckRGY, you may want to start it by hand.


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

```
rgy_test_driver: Running CASE 1
DATE:     20     13     13     25
procs    memory              page               faults        cpu
----- ----------- ------------------------ ------------ -----------
 r  b   avm   fre  re  pi  po  fr   sr  cy  in   sy  cs us sy id wa
 0  0  7042   341   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     0:37   16  1260  3376 32768  1465  2028  2.5  5.0 secd -bootstrap
 11437      - S     0:22   26   868  1064 32768   115   128  1.5  2.0 rpcd
 12327      - S     0:29   37  1408  2496 32768   745  1020  2.0  4.0 cdsd -a
 13345      - S     0:33   55  1240  1640 32768   363   324  2.3  3.0 /opt/dcelocal/bin/
cdsclerk -U root -u 0 -m 8198
 17826      - S     0:06   31   900  1220 32768   284   244  0.5  2.0 dtsd
 19017      - S     0:05    1   868  1384 32768   292   440  0.4  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 168
-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      31245 Nov 20 13:03 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
rgy_test_case: Output file is /tmp/rgy.test.15385
Current site is: registry server at /.../nashua.cell/subsys/dce/sec/master
Domain changed to: org
Domain changed to: group
Domain changed to: principal
Domain changed to: account
bye.
```

```
rgy_test_driver: Running CASE 2
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
rgy_test_case: Output file is /tmp/rgy.test.15437
Current site is: registry server at /.../nashua.cell/subsys/dce/sec/master
Domain changed to: org
Domain changed to: group
Domain changed to: principal
Domain changed to: account
bye.


rgy_test_driver: Running CASE 3
DATE:      20      13      17      54
...
```

Appendix G: Listing of the `CheckRGY` 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 Registry 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 Registry response time every $DURATION seconds
to $OUTPUTFILE ..." > /dev/tty

CheckDCE()
{
        echo -n "DATE: " >> $OUTPUTFILE
        date "+%d %H %M %S" >> $OUTPUTFILE
        rgy_edit -v -a "cell_admin"  >> $OUTPUTFILE
}

while :
do
        # What time is it now: date hour24 minute second
        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`

```
•••
DATE: 11 00 49 49
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 2 seconds.
DATE: 11 00 50 51
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 3 seconds.
DATE: 11 00 51 53
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 3 seconds.
DATE: 11 00 52 55
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 104 seconds.
SLEEP ERROR: This log ended after next one should begin
DATE: 11 00 54 38
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 98 seconds.
SLEEP ERROR: This log ended after next one should begin
DATE: 11 00 56 17
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 72 seconds.
SLEEP ERROR: This log ended after next one should begin
DATE: 11 00 57 30
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 5 seconds.
DATE: 11 00 58 29
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 5 seconds.
DATE: 11 00 59 30
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 5 seconds.
DATE: 11 01 00 32
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 4 seconds.
DATE: 11 01 01 32
Current site is: registry server at /.../tulsa.cell/subsys/dce/sec/master
cell_admin [none none]:*:100:12::/::
ELAPSED time is: 2 seconds.
•••
```